

EXPERT INSIGHT

NetSuite for Consultants

Expert strategies to streamline and
optimize implementations in
NetSuite 2025.1

Third Edition

Peter Ries



<packt>

NetSuite for Consultants

Third Edition

Expert strategies to streamline and optimize implementations
in NetSuite 2025.1

Peter Ries



NetSuite for Consultants

Third Edition

Copyright © 2025 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Portfolio Director: Pavan Ramchandani

Relationship Lead: Alok Dhuri

Program Manager: Divij Kotian

Content Engineer: Akanksha Gupta

Technical Editor: Vidhisha Patidar

Copy Editor: Safis Editing

Proofreader: Akanksha Gupta

Indexer: Tejal Soni

Production Designer: Nilesh Mohite

Growth Lead: Nivedita Singh

First published: January 2022

Second edition: May 2023

Third edition: August 2025

Production reference: 1120825

Published by Packt Publishing Ltd.

Grosvenor House

11 St Paul's Square

Birmingham

B3 1RB, UK.

ISBN 978-1-83664-077-6

www.packtpub.com

I would like to dedicate this book to all of the dreamers, seeking a better life.

– Peter Ries

Contributors

About the author

Peter Ries is a consulting technical director at Oracle NetSuite. He has been providing high-quality consulting for innovative software solutions for many years.

He enjoys solving tough business problems, developing new software tools, and integrating applications to make his clients' businesses run on NetSuite. He works with a variety of current technologies and loves working with all levels of people within each organization. When he is not working, he spends time with his family, reads and writes, and rides bikes as much as time allows.

I would like to thank my family for their constant love and support, Ilyaad Damree, for his expert reviews and advice, and the wonderful team at Packt Publishing, without whom this book would not exist.

About the reviewer

Ilyaad Damree is an entrepreneur, full-stack developer, and NetSuite consultant with over a decade of experience. Blending technical expertise with business insight, he has played both technical and functional roles in various NetSuite projects. His versatility has taken him to multiple implementation projects across Western Europe, the Nordics, and North America. In 2023, he solidified his position as a solution architect by earning all seven NetSuite certifications. Ilyaad is passionate about sharing his knowledge to support learning within the NetSuite community.

Thank you, Peter, for the opportunity to review this work—it was a chance to learn more than I expected. I'm also grateful to my team at NX2SQUARE, and to all the colleagues, past and present, who've inspired, challenged, or unintentionally taught me something along the way. You know who you are.

Table of Contents

Preface

xvii

Your Book Comes with Exclusive
Perks - Here's How to Unlock Them xxi

Part 1: The NetSuite Ecosystem, Including the Main Modules, Platform, and Related Features

1

Introduction to the NetSuite Ecosystem, Platform, and Related Features

3

NetSuite the company, NetSuite the product	4	SuiteBuilder	16
NetSuite editions, features, accounts, and centers	4	SuiteScript	16
NetSuite accounts	5	SuiteFlow	16
How NetSuite is updated	8	SuiteTalk (SOAP and REST)	17
NetSuite people – sales, support, and services – and partners	10	SuiteAnalytics Connect	17
NetSuite's product offerings	11	NetSuite's mobile apps	18
The NetSuite SuiteCloud Platform	15	Summary	19
		Self-assessment	19
		Learn more on Discord	19

2

Selecting and Applying an Implementation Methodology

21

What is a methodology and why do we need one?	22	Selecting from the different approaches	23
---	----	---	----

The Waterfall method	24	Starting with Waterfall, applying Agile when necessary	30
The Agile method	26		
Adapting the methodology to each client	30	Summary	32
		Self-assessment	32

3

Creating a Project Plan 33

What goes into a project plan?	33	Applying your methodology to the project plan	39
Setting project goals	34		
Defining the tasks in the project plan	36	Living with the plan as changes occur	41
Conference room pilots, walk throughs, and testing	37	Summary	42
		Self-assessment	43

Part 2: Understanding the Client's Organization

4

Documenting the Organization's Requirements 47

Understanding the organization's industry	48	Use case – differences between a wholesale distributor and a software company	54
Manufacturing	48	Gathering requirements and interviews	57
Not for profit	48	Project kickoff	58
Retail	49	Business process reviews	58
Software	49	Demos, conference room pilots, and walk-throughs	58
Services	49	UAT	59
Wholesale/distributor	50	Go-live	59
Miscellaneous	50		
Understanding the organization's business and people	51	Documenting requirements	60
Project manager	52	Summary	62
The subject matter experts	53	Self-assessment	62
The users/testers	53		
Other project helpers	54		

5

Analyzing the Organization's Users and Roles 65

Understanding departments, teams, managers, and users	66	Which custom records does the role need access to?	69
Accounting	66	Which reports will the role need access to?	69
Accounts payable	66	What do you want the role's dashboard to have on it?	70
Accounts receivable (A/R)	66	Which forms and searches should be the role's preferred versions?	70
Administrators/upper management	66	Creating and managing employees	71
Information technology	67	Grant access	72
Sales	67	Assign roles	72
Support	67	Set up their supervisor	72
Warehouse	67	Set up their expense limits and approvals	73
Grouping people with roles in NetSuite	67	Managing users with multiple roles	73
Does the user need to view, create, edit, or fully manage a given record or list?	69	Summary	73
		Self-assessment	74

6

Understanding the Organization's Accounting and Finance 75

Enabling features and setting up the accounting basics	76	Understanding the Accounts Payable feature	83
Defining the general ledger and Chart of Accounts	78	Understanding the Accounts Receivable feature	83
Setting up segmentation—subsidiaries, locations, departments, and classes	80	Use budgeting to plan and control finances	84
Subsidiaries	80	Special use case – fixed asset management	85
Locations	80	Summary	86
Departments	81	Self-assessment	86
Classes	82		
Custom segments	82		

7

Getting to Know the Organization's Entities and Items 89

Differentiating client leads, prospects, and customers	90	Managing vendors, partners, and other entity types	94
Implementing customer projects and resources	92	Identifying items and their types	96
When to use contacts versus sub-customers	93	When we use special-purpose items	98
		Summary	100
		Self-assessment	100

8

Identifying the Organization's Main Transactions 101

Grouping records by business process	101	Lead to quote	107
Gathering requirements for RTR transactions	103	Call to resolution	107
Gathering requirements for PTP transactions	104	Hire to pay	107
Gathering requirements for OTC transactions	105	Marketing to return on investment	107
Analyzing the other process groups	106	Revenue recognition	108
Design to build	107	Return to credit	108
		Project to delivery	108
		Web to order	108
		Summary	109
		Self-assessment	109

Part 3: Implementing an Organization in NetSuite

9

Custom Forms, Records, and Fields 113

What is SuiteBuilder and when should you use it?	114	Removing fields the users do not need to see or will not use	117
Customizing entry and transaction forms	115	Adding fields the users need that were not included out of the box	118
		Rearranging the fields and lists on the form	118
		Enabling related features and settings	118

Storing the form with the record	120	Image versus inline HTML	124
Actions	120	Change management for all your custom objects	124
Roles	120	Requirements or use cases	125
Using custom record types to store additional non-standard data	120	Issues or defects	125
Defining custom lists and fields	122	System change	125
Checkbox versus list	123	Summary	126
Single-select lists versus multiple-select lists	123	Self-assessment	126

10

Centers and Dashboards 129

Setting up native centers	130	Use case – setting up and using the Customer Center	135
Customer Center	131	Setting up dashboards for groups by role	136
Employee Center	131	Summary	138
Partner Center	132	Self-assessment	139
Vendor Center	132		
Using custom centers for even greater control over the UI	134		

11

Items and Related Lists 141

Enabling item types, inventory management options, and so on	141	Assembly items versus item groups versus kits	148
Defining the item forms and fields	143	Other charges	149
Defining purchase prices and sales pricing for items	145	Subtotal	149
Setting up matrix item types and item options	146	Special feature: Supply Chain Control Tower	149
Setting up other item types as needed	147	Summary	151
Description	147	Self-assessment	152
Discount	147		

12

Customers, Vendors, Contacts, and Other Entities 153

How to set up customers in the account	154	Multiple subsidiaries	159
Duplicate records	156	Addresses	159
Bad addresses/phone numbers	156	Financial considerations	159
Records that are otherwise invalid	156	Vendor bill/three-way matching	160
Do you need contacts, and should you create them?	158	Setting up competitors, partners, and other entities	161
Defining vendors for positive PTP processes	159	Managing projects and resources	161
		Summary	164
		Self-assessment	164

13

Financial Transactions and Period Closes 167

Setting up and using the GL	168	Elimination	176
Implementing currency management	169	Statistical journals	176
Customer and vendor currency lists	170	Performing and managing period closes	177
Exchange rates for multiple currencies	170	Manage accounting period	178
Setting up and using banking features	172	Override period restrictions	178
Using Journal Entries	173	Overview of NetSuite's tax features	179
Reversals	174	Summary	180
Advanced versus normal intercompany journals	175	Self-assessment	181
Revenue recognition	175		

14

Procure-to-Pay Transactions 183

Using requisitions, purchase requests, and purchase orders	184	Utilizing inbound shipments and processing item receipts	187
Managing special orders and drop-ship orders	186	Planning for and using vendor bills	189

Tracking vendor prepayments and making vendor payments	190	Processing vendor returns and vendor credits	192
		Summary	193
		Self-assessment	194

15

Order-to-Cash Transactions 195

Using opportunities and estimates before sales	195	Managing Item Fulfillments for sales, transfers, and returns	201
Recording sales via sales orders and cash sales	197	Making the most of invoices	203
Bulk/blanket orders and order guides	198	Entering Customer Deposits and Payments	204
Drop-Ship Orders/Special Orders/Special Work Orders	199	How to track Return, Credit, and Refund transactions	207
E-commerce sales orders	199	Summary	208
Project/service deliverables	199	Self-assessment	208
Promotions	200		
Revenue recognition	200		

16

Other Transactions and Custom Transactions 211

Using work orders and assembly builds for manufacturers	212	Revenue arrangements	219
Assembly items	212	Revenue allocation	219
Work orders	213	Amortization	219
Assembly builds	213	Using Cases and Issues for customer support	220
WIP/Manufacturing Routing	214	Creating custom transactions for special sales and purchases	222
Managing revenue recognition with ARM	216	Summary	223
Advanced Revenue Management	217	Self-assessment	223
Revenue recognition rules	218		
Revenue recognition plans	219		

17

Analytics, Reports, and Data Exports 225

Using searches and views to find information	226	Linking datasets for even more flexibility	235
Global search	226	Creating exports with SuiteAnalytics Connect	236
Saved searches	227	Summary	239
Creating reports for in-depth analysis	229	Self-assessment	240
Utilizing datasets and workbooks for deeper analysis	232		

Part 4: Managing Gaps and Integrations

18

Managing Gaps and Creating Custom Automations 243

Identifying gaps and needs for automation in requirements	244	Scheduled and Map/Reduce scripts	252
Customizing NetSuite processes with workflows	245	Suitelets and RESTlets	252
Making NetSuite work for you with scripts	248	Documenting customization requirements and solutions	253
Client scripts	251	Managing customizations over time, including their deployments	255
User Event scripts	251	Summary	257
		Self-assessment	257

19

Managing Integrations 259

Talking to clients about integrations	260	Option 1 – SuiteScripts	265
Talking about data preparation for integrations	261	Option 2 – SuiteTalk SOAP	266
Collecting information about and documenting integrations	262	Option 3 – SuiteTalk REST	266
Managing data coming into NetSuite	264	Option 4 – Middleware	266
		Option 5 – SuiteScript RESTlet	267

Creating custom exports from NetSuite	267	Summary	269
		Self-assessment	269

20

Managing Data Migrations 271

Planning for the data migration process	272	Migrating inventory and trial balances	278
Performing imports for data migration testing and other reasons	274	Performing the final data imports before going live	278
Testing the data in phases	276	Summary	279
Data migration performance	276	Self-assessment	280
Planning the cut-over activities	277	Learn more on Discord	281

Appendix

My Answers to the Self-Assessments 283

Chapter 1	283	Chapter 11	289
Chapter 2	284	Chapter 12	289
Chapter 3	284	Chapter 13	290
Chapter 4	285	Chapter 14	290
Chapter 5	285	Chapter 15	291
Chapter 6	286	Chapter 16	291
Chapter 7	287	Chapter 17	292
Chapter 8	287	Chapter 18	293
Chapter 9	288	Chapter 19	293
Chapter 10	288	Chapter 20	294

Index 297

Other Books You May Enjoy 308

Preface

NetSuite was created in 1998, in Northern California, from a desire to bring small- and medium-sized business processing into the internet age. A few people who worked for Oracle at the time wanted to create a Relationship Management (CRM) and Enterprise Resource Planning (ERP) product you could run in your web browser without ever having to install anything in your offices. They received seed money for this new company and embarked upon their 20+ year journey to making the set of products we use today.

I began working for NetSuite in 2012 as a technical consultant, delivering customizations and integrations for clients who were just about to go live on the platform. I've been working at that ever since, and one thing I've learned is that every business is unique and brings its own challenges when it comes to implementing NetSuite. At the same time, however, all companies share a few common traits when it comes to introducing them to the NetSuite line of products.

With this book, I hope to share what I've learned about helping clients use NetSuite. I'd like to think I can help make the process of getting a company up and running on NetSuite less painful and faster than it might otherwise be. I hope you find this useful and will engage with me and the rest of the NetSuite user community in online discussions regarding this topic.

Who this book is for

This book has two intended audiences. One is people who work for a company that helps other businesses implement NetSuite. We generally refer to those companies as NetSuite partners or vendors, or sometimes system integrators. They have lots of people on staff whose job is to work with NetSuite client companies and help them get started on the platform. These folks need to learn more about NetSuite and how to work with clients.

The other audience for this book is anyone working at a NetSuite client company who needs to help their users do more with the product – or maybe they're considering whether they could self-implement without the help of a partner. This will always be a large, challenging task, but you might feel more confident about taking this on after reading this book.

What this book covers

Part 1: The NetSuite Ecosystem, Including the Main Modules, Platform, and Related Features

Chapter 1, Introduction to the NetSuite Ecosystem, Platform, and Related Features, starts with an explanation of what NetSuite is all about and what you can do with it.

Chapter 2, Selecting and Applying an Implementation Methodology, covers implementation project methods and how to select the right one for every client.

Chapter 3, Creating a Project Plan, discusses what a NetSuite implementation project consists of and how you can schedule your time.

Part 2: Understanding the Client's Organization

Chapter 4, Documenting the Organization's Requirements, covers all of the challenges of gathering requirements and techniques you can use to streamline the process, which is the most important step at an early stage in every implementation.

Chapter 5, Analyzing the Organization's Users and Roles, explains how people make the system work, and so we need to understand who they are and what they will do in the system.

Chapter 6, Understanding the Organization's Accounting and Finance, delves into the accounting and finance functions in a NetSuite account to help you learn how to work with these users to understand their needs.

Chapter 7, Getting to Know the Organization's Entities and Items, talks about gathering requirements relating to the business entities, items, and projects and how important they are to the rest of the implementation process.

Chapter 8, Identifying the Organization's Main Transactions, covers the transactions that are a day-to-day operational need for most businesses. You'll learn how to talk to clients about the transactions they will use the most.

Part 3: Implementing an Organization in NetSuite

Chapter 9, Custom Forms, Records, and Fields, moves on from requirements gathering to the first steps we usually take when configuring an account to work as the client needs it to.

Chapter 10, Centers and Dashboards, covers how, given that organizing the data and activities in an account for users is an important task for the implementation team, you can set up the home screen and other areas for maximum efficiency.

Chapter 11, Items and Related Lists, details how to help a client get items and related lists set up correctly the first time, whether you're tracking inventory items or services.

Chapter 12, Customers, Vendors, Contacts, and Other Entities, covers how the entity lists in NetSuite allow us to keep track of all the other companies we do business with. Getting them set up well requires a solid understanding of your client's requirements plus the native NetSuite features.

Chapter 13, Financial Transactions and Period Closes, covers how to configure and train your users on financial transactions – journal entries and such.

Chapter 14, Procure-to-Pay Transactions, covers purchasing transactions and everything you need to know about all of the options available in NetSuite.

Chapter 15, Order-to-Cash Transactions, covers all the native transactions in the order-to-cash business process.

Chapter 16, Other Transactions and Custom Transactions, addresses a couple of special use cases to explain how we handle unusual conditions when we need to.

Chapter 17, Analytics, Reports, and Data Exports, goes into all of the amazing options we have for reporting on our data within the product, including information on SuiteQL.

Part 4: Managing Gaps and Integrations

Chapter 18, Managing Gaps and Creating Custom Automations, covers NetSuite's SuiteCloud platform, and all of the amazing customizations and automation we can create with it to fine-tune the suite to work for each client.

Chapter 19, Managing Integrations, explains the basics of integrations, which allow us to bring outside data into the system, or export our NetSuite data elsewhere, although a lot of experience is required to know how to handle this well.

Chapter 20, Managing Data Migrations, explains how to plan for a migration, test it to ensure your success, and pull it off as part of the client's go-live process.

Appendix, My Answers to Self-Assessments, provides suggested answers to the questions at the end of each chapter.

To get the most out of this book

You should have access to a NetSuite account in order to follow along with each chapter. Access to the NetSuite Help system and SuiteAnswers, or the public documentation available at <https://docs.oracle.com>, will help a lot as well. Note: Every NetSuite account has different features enabled within it, so your account may not always give you access to all of the features shown in this book. When you can't use a feature in your account, I hope the book explains its use sufficiently clearly.

After you've read this book, please visit the companion website at <https://implementingnetsuite.com/> to follow the conversation or reach out to me.

Conventions used

There are a number of text conventions used throughout this book.

- **Bold:** Indicates a new term, an important word, or words that you see onscreen. For instance, words in menus or dialog boxes appear in **bold**. Here is an example: “Apply preferences, such as how PDFs will be handled or whether **inventory-level warnings** will be displayed, via the user’s **Preferences** screen.”
- `CodeInText`: Indicates database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and X/Twitter handles. For example: “As you are helping a client import their `Employee` list for the first time, the client asks whether they can attach a custom record to employees.”

Tips or important notes

Appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, email us at customercare@packtpub.com and mention the book title in the subject of your message.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/support/errata and fill in the form.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.


Your Book Comes with Exclusive Perks - Here's How to Unlock Them

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search this book by name. Ensure it's the correct edition.

Note: Keep your purchase invoice ready before you start.

UNLOCK NOW














Figure 0.1: Next-Gen Reader, AI Assistant (Beta), and Free PDF access

Enhanced reading experience with our Next-gen Reader:

 **Multi-device progress sync:** Learn from any device with seamless progress sync.

 **Highlighting and notetaking:** Turn your reading into lasting knowledge.

 **Bookmarking:** Revisit your most important learnings anytime.

 **Dark mode:** Focus with minimal eye strain by switching to dark or sepia mode.

Learn smarter using our AI assistant (Beta):

✦ **Summarize it:** Summarize key sections or an entire chapter.

✦ **AI code explainers:** In the next-gen Packt Reader, click the **Explain** button above each code block for AI-powered code explanations.

Note: The AI assistant is part of next-gen Packt Reader and is still in beta.

Learn anytime, anywhere:



Access your content offline with DRM-free PDF and ePub versions—compatible with your favorite e-readers.

Unlock Your Book's Exclusive Benefits

Your copy of this book comes with the following exclusive benefits:



Next-gen Packt Reader



AI assistant (beta)



DRM-free PDF/ePub downloads

Use the following guide to unlock them if you haven't already. The process takes just a few minutes and needs to be done only once.

How to unlock these benefits in three easy steps

Step 1

Keep your purchase invoice for this book ready, as you'll need it in *Step 3*. If you received a physical invoice, scan it on your phone and have it ready as either a PDF, JPG, or PNG.

For more help on finding your invoice, visit <https://www.packtpub.com/unlock-benefits/help>.

Note: Did you buy this book directly from Packt? You don't need an invoice. After completing Step 2, you can jump straight to your exclusive content.

Step 2

Scan this QR code or go to [packtpub.com/unlock](https://www.packtpub.com/unlock).



On the page that opens (which will look similar to Figure 0.2 if you're on desktop), search for this book by name. Make sure you select the correct edition.

The screenshot shows the Packt website's 'Discover and unlock your book's exclusive benefits' page. At the top, there's a navigation bar with the Packt logo, a search bar, and links for Subscription, Shopping Cart, and User Profile. Below this is a secondary navigation bar with links: Explore Products, Best Sellers, New Releases, Books, Videos, Audiobooks, Learning Hub, Newsletter Hub, and Free Learning. The main heading is 'Discover and unlock your book's exclusive benefits', followed by a subtext: 'Bought a Packt book? Your purchase may come with free bonus benefits designed to maximise your learning. Discover and unlock them here'. A progress bar shows three steps: 'Discover Benefits' (active), 'Sign Up/In', and 'Upload Invoice'. Below the progress bar, there's a section titled '1. Discover your book's exclusive benefits' with a search bar labeled 'Search by title or ISBN' and a 'CONTINUE TO STEP 2' button. Below this is a section titled '2. Login or sign up for free'.

Figure 0.2 - The webpage to help you unlock your book's benefits.

Step 3

Sign in to your Packt account or create a new one for free. Once you're logged in, upload your invoice. It can be in PDF, PNG, or JPG format and must be no larger than 10 MB. Follow the rest of the instructions on the screen to complete the process.

Need help?

If you get stuck and need help, visit <https://www.packtpub.com/unlock-benefits/help> for a detailed FAQ on how to find your invoices and more. The following QR code will take you to the help page directly:



Note:

If you are still facing issues, reach out to customercare@packt.com.

Share Your Thoughts

Once you've read *NetSuite for Consultants*, we'd love to hear your thoughts! Please [click here](#) to go straight to the Amazon review page for this book and share your feedback.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content.

Part 1:

The NetSuite Ecosystem, Including the Main Modules, Platform, and Related Features

In this part, we will understand what NetSuite is, how it works, and how accounts, support, updates, and so on work within its ecosystem. We will also learn how to select an implementation project methodology and how to plan the work within a project.

This part has the following chapters:

- *Chapter 1, Introduction to the NetSuite Ecosystem*
- *Chapter 2, Selecting and Applying an Implementation Methodology*
- *Chapter 3, Creating a Project Plan*

Introduction to the NetSuite Ecosystem, Platform, and Related Features

Welcome to *NetSuite for Consultants*. In this book, I assume that you already know about NetSuite's **enterprise resource planning (ERP)** and **customer relationship management (CRM)** products, and I will cover everything you should know when starting to implement the product for your own company, or as a solution provider or consultant. I'll explain the soft skills a consultant should have, how to get to know the client's business and people, how to analyze and document client requirements, and much more. We'll then dig into the process we follow when implementing NetSuite and break that down by business process. Finally, I'll cover how to handle *gaps* you find between the client's requirements and NetSuite's features, so you know how to handle whatever comes your way.

To implement NetSuite, you've got to understand a lot about NetSuite as a company, the NetSuite product, the customization platform, and so much more. I'll provide some of those details in this chapter before we dig into the remaining details in the rest of the book.

In this chapter, we're going to cover the following topics:

- NetSuite the company, NetSuite the product
- NetSuite editions, features, accounts, and centers
- NetSuite people – sales, support, and services – and partners
- The NetSuite SuiteCloud Platform and the mobile apps

This chapter will introduce you to all those topics and more, ensuring you get a great start down the right path to NetSuite consulting.

NetSuite the company, NetSuite the product

NetSuite is both a company and a product. The company was founded as *NetLedger* in 1998 by ex-Oracle employees, with the dream that someday, a complete ERP system running in the cloud would be a good fit for many small and mid-sized companies. At that time, only larger enterprises were offered a single application offering – everything a company would need to run its business. Smaller companies had to piece together applications such as accounting, inventory management, and billing themselves, with the help of system integrators. The accounting software of the day was either like QuickBooks, which ran on a personal computer with a local database, or a much more complicated server application with a relational database, which required a team of IT specialists to set up and maintain. When companies outgrew QuickBooks, they had to move up to the larger, on-premises-installed applications. The market was not geared toward mid-sized companies at that time, though options from Microsoft and others did fit the bill for some of the main ERP functions.

NetSuite's founders' vision was to build a web-based application to replace those *on-premises* applications and, in the process, relieve their customers of all the IT headaches that come with running servers as well as installing, maintaining, and integrating many complex business applications. And so NetSuite was born!

NetSuite initially offered the simplest options for basic company accounting and customer record management, under the name NetLedger.com. It has grown over the years to truly be a massive suite full of useful options for nearly every type of business and non-profit organization. In 2017, Oracle acquired NetSuite and the product has continued to grow and cover more ground since then.

Since NetSuite is cloud-based, you can access it from almost any internet-connected device with a web browser. With your PC or tablet (or maybe even your phone), you can log in and do whatever you need in your NetSuite account to run your business. NetSuite maintains many data centers around the world, thanks to Oracle's expertise and reach in this industry. Whether your employees are all in one city, or are spread across continents, they can easily share one common interface, any day of the week. NetSuite's *multi-tenant* approach to cloud computing means that you never have to worry about which server your account is running on; you trust that NetSuite is making sure that your account will always be online, 24 hours a day, every day.

NetSuite editions, features, accounts, and centers

Oracle NetSuite (the company) has built the NetSuite products so that clients don't need to worry about which server their NetSuite account is running on. When a business first signs up for the service, the NetSuite sales team will size up the organization and help them choose the right package of options. This includes getting them on the correct server tier. A basic starter account for a smaller company typically runs on a shared cloud-based application server platform so that computing resources are efficiently shared among a set of companies. However, the data that's accessed by each is completely segregated, so each company will only ever have access to its own records. Companies that require greater performance can purchase premium server tiers and can even request dedicated servers when

the need arises. This is typically done for companies with high transaction volumes who cannot afford to ever be slowed down by another company's activities on a shared server account. Read the NetSuite Help page titled *NetSuite Service Tier Structure* for more on this: https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/sect_159162378384.html.

Each NetSuite client will typically have access to one NetSuite account, which is a separate instance of the NetSuite system they can work in. Some companies do require multiple accounts, though, for extremely high transaction volumes or when other complex requirements exist. Next, we'll cover the different types of NetSuite accounts and how each can be used: Production, Sandbox, Development, and **Release Preview (RP)**.

NetSuite accounts

When an organization signs up for NetSuite, they will always start with one main **Production** account. All NetSuite applications run in the cloud and all users will access their account via a URL such as <https://1234567.app.netsuite.com>. This is where they are expected to run their business, once they're ready to. There are other types of accounts available, though, depending on their needs.

The following diagram shows the account types that are available at the time of writing:



Figure 1.1: NetSuite account types

Let's briefly talk about each of them.

Production accounts

Most NetSuite clients have just one of these but it's possible to ask for more under special circumstances. This is where your data will primarily live, where emails are sent and integration connections are made. For most of a client's users, this is NetSuite as far as they are concerned. For that reason, the data in these accounts is critical to the organization and should be treated as such.

Sandbox accounts

Think of these as temporary accounts where the development of customizations and any testing should occur, so these activities won't cause any problems with the production data. NetSuite sells access to these accounts for an extra fee. Sandboxes are *refreshed* from the Production account on an *as-needed basis* and, when that occurs, they then have a separate database reflecting the changes that were made in the **Sandbox** account. One way to move things from a sandbox's configuration to a Production account is via the **Copy to Account** feature, which we'll talk a bit about in *Chapter 18*.

One other big difference with Sandbox accounts is that they don't send emails to their intended addressee. Instead, they are redirected either to a set of named users or to the NetSuite user who was logged in when the email was generated. Otherwise, NetSuite will not send emails out from a Sandbox account.

Development accounts

With each sandbox a client purchases, NetSuite provides three additional accounts of this type. They have a separate database as well, but they are never refreshed from Production. Instead, they are only used to develop standalone customizations that can then be deployed into any of the other accounts for testing or production use.

These **Development** accounts can be very helpful for companies who are developing customizations, such as scripted automations or integrations. For almost any other purpose, they can be more trouble than they're worth since they can't automatically get data from Production and you have to manually set up everything you want in them, such as bundles, SuiteApps, and other automations or customizations.

Release Preview accounts

Just before NetSuite's biannual major updates are released, the company offers each client a copy of their Production account for advanced testing of the new features.

Each NetSuite client must request their RP account to be set up, but there is no additional fee for this service.

These RP accounts only last for a few weeks, though – just enough time to test new features and confirm that existing configurations and customizations will continue to operate as expected.

Work with your NetSuite account manager whenever you want to make a change to any of a client's accounts.

Data in NetSuite accounts

NetSuite maintains daily internal backups of every account. However, no client should ever assume that the backup is going to be there to cover any mistakes they might make or, worse, to recover from failed experiments. Instead, a carefully planned data management approach is needed. Most clients can run their accounts for years without any incidents, so they won't need to restore a backup, which is largely different from most on-premises software applications. This is because NetSuite (the company)

takes such care to make sure the application runs without errors or other problems causing any harm to the data. There are, however, many ways a client or users can cause issues themselves. For instance, NetSuite includes two features, known as **CSV Import** and **Mass Updates**, which are very powerful but also very dangerous if they're used incorrectly or carelessly.

I've seen more than a few cases where someone ran CSV Import and overwrote important data incorrectly or inadvertently. The same goes for Mass Updates. Generally, these features are restricted to NetSuite administrators and should stay that way, but if a client gives an inexperienced user (including consultants like us) the power to use any of these powerful features, bad things can happen.

A client can contact NetSuite Support and request data to be restored, but that should be left as a last-resort option. Instead, any time anyone is thinking of making any sort of bulk update to data, or maybe changing data they're not very familiar with, it's a best practice to make a manual backup of that data first, via a CSV export. We generally should not rely on NetSuite to provide backups, but you may ask them for help of this type in emergency cases.

This situation is mainly for the data in Production accounts, but there are times when the data in Sandbox accounts can be just as critical. In a typical implementation, we might import lists of things such as customers and items fairly early on in the project, and once those are in place, we don't want anyone erasing them or making large-scale changes without careful planning. Having to re-import a lot of data midway through a project can introduce unplanned delays, if not done correctly. Also, keep in mind that when a sandbox is refreshed, all of its current data is erased/lost and replaced with the data that was found in the Production account (or another Sandbox account) at that time. Clients and implementation teams need to carefully review the data in the sandbox before these refresh events, to be sure nothing important in the account will be lost.

Sandbox refreshes

A typical mid-sized or larger NetSuite client, when going through an implementation process, will usually end up following a schedule that looks something like this:

1. **Day 1:**

- Start to configure the Production account
- We set up the first users, enable features, and start to choose options
- Frequently, we also perform initial configurations of other things in Production at this stage, such as the **Chart of Accounts (COA)** and items

2. **About a month or so later:**

- Request a refresh for the first sandbox
- The client usually does this once they've gotten the OK from all the teams working in the Production account

- This is where customizations, integrations, and data migrations will usually be developed and tested (although this can happen in the Production account on simpler implementations instead)

3. **Sometime later:**

- Request a refresh for a second sandbox or the first sandbox, if only one is available
- This is where **user acceptance testing (UAT)** will happen, before going live, so this can only happen after all the elements needed for that full round of end-to-end testing have been configured or installed

4. **Days before going live:**

- Deploy all customizations to Production, migrate the final data, and configure the integrations
- Sandboxes cannot be refreshed into a Production account, so we usually use other deployment features at this time, such as suite bundles and the **SuiteCloud Development Framework (SDF)**
- Then we perform the other cut-over activities (such as last-minute training or updating the business process documentation)

We'll cover this in more detail later in this book but keep this general process in mind as you continue reading. Note that the actual process we follow is dependent on the size and complexity of the organization that's being implemented. Many smaller companies can get through their entire implementation process with just their Production account, for instance, which greatly simplifies and speeds up this process, but larger clients may require more sandboxes.

How NetSuite is updated

Since NetSuite runs in the cloud and not in every client's office, it is NetSuite's job to keep the application software up to date. The product receives updates automatically, twice per year, typically around April and November. The first of these *major* updates is known as the **.1** release for that year, while the second is known as **.2**. So, this year, at the time of writing, clients will be updated to *2025.1* in April and *2025.2* in November. These updates include both fixes for reported problems and new features and enhancements to the product.

In between these two major updates, NetSuite can also patch any part of the system via what are known as either e-fixes or hot pushes. An **e-fix** is a small update that's used to fix defects reported by NetSuite clients and rolled out to the live systems only when needed. **Hot pushes** are very rare as they are emergency patches needed by one or more clients, usually to resolve a recently introduced product defect. For instance, this kind of update might be used if a problem with a major release causes issues for many clients simultaneously.

NetSuite clients cannot choose to never receive these updates; however, they can sometimes choose when they receive them and which of their accounts will be updated first. When this is needed, the client can work with NetSuite Support to request alternatives to the default rollout plan.

Whenever a major update is about to be released, NetSuite's clients and their partners should start to plan for how to adapt to the new features and incorporate them into their business process. For instance, on occasion, a new feature is released that can replace a customization a client had developed in the past. When this happens, the client should evaluate the new feature, compare its setup, records used, and so on to their custom solution, and then define a plan for how the business will migrate into the now native feature (assuming that makes sense). Generally speaking, it's usually better for a client to run with as few custom solutions as possible, so this kind of thing is usually the right choice. This is where using those RP accounts mentioned earlier can be very helpful, although it's always important to keep their temporary nature in mind. Don't ever start a development project in an RP account!

Since major updates are automatically applied to each account twice per year, nothing special ever needs to be done by the client to enable/facilitate this. The upgrade usually happens overnight (depending on where the client is); when you first sign in the next day, the account's version number will have changed. The same goes for hot pushes, except the minor revision version number indicating a change at that level is not displayed on the home page.

At that point, conscientious clients (and partners, among others) will perform *smoke* testing for their main business processes to ensure that everything works as expected. It is very rare for issues to arise because of an upgrade, but it's always better to find them via testing than to find out after a week or more that some critical function is no longer delivering the expected result.

You can find the revision number; it's just not in an obvious place. Sign in to NetSuite, right-click the home page in your web browser, and select **View Page Source**. Scroll to the very end of that source listing and look for a block of text that looks like this:

```
<!-- Host [ abcdef ] App Version [ 2025.2.0.96 ] -->
<!-- COMPID [ 1234567 ] URL [ /s.nl ] Time [ Sun Jan 10
15:36:03 PST 2023 ] -->
<!-- Not logging slowest SQL -->
2025.2.0.96
```

In this example is the full version number of the account you're looking at. That's generally not a number anyone needs to keep track of, but now and then, it can be useful to know how to find this information.

Generally, NetSuite is very, very careful to ensure that every update they release is well tested and that it will not break any existing configuration or customization clients might have. When the company knows that a change is going to cause issues, they almost always provide many months' worth of advanced notice so that clients can prepare their business and any integrations to support the new approach. This is especially true for things such as integration protocols, such as SAML for single sign-on, or various features of the SuiteCloud platform. In my many years of working on the product, I've only seen three or four cases personally where an update caused an issue and the client wasn't given months of notice before the change took place.

Now that we've seen what NetSuite is, let's look at the people who work to make NetSuite the incredible product it is – and how they can help you do your job too.

NetSuite people – sales, support, and services – and partners

NetSuite was founded in northern California but is now a global organization. Oracle's headquarters moved from California to Austin, Texas, at the beginning of 2021, but NetSuite's people reside in many countries around the world. You'll find NetSuite employees based in North America, and there are teams in Australia, Europe, Asia, and Central and South America too. Salespeople are the frontline points of contact, helping new clients get started on the product.

The sales teams are divided into two groups – direct sales and account managers. The direct sales team gets you started and the account managers take over just before an organization *goes live*.

NetSuite's Customer Success teams offer professional services for any client looking for help with their implementation (or any time after that, really) but NetSuite also sells services from third parties just as often. These *partners* have many well-trained and experienced professionals, such as in-house consultants, and again, they can be found in all NetSuite markets and provide all of NetSuite's ERP, CRM, and such related services, from accounting and private equity consulting to software development.

NetSuite's Education Services and Support people are some of the most patient and knowledgeable people you can work with. While the trainers are available in every market, the Support team is mostly found in the Philippines and near Toronto, Canada. NetSuite offers a variety of training and support packages but most recently has been encouraging clients to sign up for the *always available* subscription services, ensuring that all of the organization's employees are well trained and can get help when they need it.

To this end, Education Services offers a **Learning Cloud Support** subscription, which means your employees can typically receive online training on any NetSuite-related subject whenever they need it (not just in the time right before going live, in other words). The Customer Success team offers something similar called **Advanced Customer Support**, which extends their basic tech support option to include dedicated resources that know your account and business and apply that knowledge to each issue you need help with. There are three levels of advanced support offered, meeting each client's needs and budget.

NetSuite's infrastructure and network people, product engineers, developers, and testers are distributed all around the globe, but most are in either the US, Canada, Spain, or the Czech Republic. Since NetSuite has acquired many other companies over the years, these developers work on a large range of products under the NetSuite umbrella.

NetSuite's product offerings

If we tried to list all of NetSuite's features and options, the list would stretch over many pages, but the product's main offerings can be summarized as follows:

- **ERP and CRM:** These products include things such as finances, procurement, and order management. This is where we'll talk about customers and contacts, items and inventory tracking, purchase and sales orders, **general ledger (GL)** accounts and journal entries, revenue recognition, and many more topics. NetSuite changes how they sell this every so often, but generally, they offer a package for small companies (those who have outgrown QuickBooks, for instance), mid-sized companies, and larger enterprises.

Most NetSuite ERP customers purchase the **OneWorld** product, which allows for things such as multiple subsidiaries, currencies, and accounting books (when needed).

The ERP and CRM features are where we will focus most of our time in this book since there is so much under this heading – we always begin new implementations with these features.

- **SuiteCommerce and MyAccount:** These are NetSuite's built-in e-commerce and customer management platforms and they include everything needed to sell products or services via the web, with all of the customer and transaction data directly tied to the same NetSuite account. (No integrations should be needed but they are supported!)

SuiteCommerce and **SuiteCommerce Advanced (SCA)** are the current basic and full-featured e-commerce offerings. **MyAccount** is a highly customizable customer management portal, allowing a business' customers to access their order history, make payments, and more.

Broadly speaking, about one-third of all NetSuite clients run their web store on the SCA platform, but many others have tied an external web store into their NetSuite accounts via integration.

This can be a very complex topic since most companies' web stores are heavily specialized and often customized, so it could easily require an additional set of chapters to cover properly. For these reasons, I will mention this subject again as appropriate, but most SCA learning will not be covered in this book.

- **NetSuite WMS:** For companies in the **Wholesale Distribution (WD)** or **Manufacturing (MFG)** industry, a **Warehouse Management System (WMS)** is key to running a highly efficient warehouse operation.

NetSuite's WMS offering includes many of the features a small to mid-sized company needs to tie their inventory and production management into their accounting systems, such as warehouse location tracking (via lots and bins, and so on), fulfillment (picking, packing, and shipping), and inventory counts.

There are a number of mobile applications available now as well, which make it easy to integrate the WMS features into your warehouse operations. These include the Manufacturing Mobile, Supply Chain, and Ship Central apps.

There are a few partners with solutions as well, and Oracle has its own WMS offering for larger operations or those with more complex warehouse requirements.

Again, exploring this topic in depth would require more space than we can provide in this book, but I will mention it a few times where appropriate.

- **The special-purpose, limited-access centers:** NetSuite also offers a set of alternative **User Interface (UI) centers** as web-based extensions for people other than the client's full-license users.
 - Customer Center allows a client's customers to directly sign in to a limited version of the NetSuite UI and place orders, review invoices, or see the status of previously placed orders.
 - Vendor Center is similarly set up for a client's vendors, enabling them to work with purchases placed by the client and to interact with them.
 - Employee Center allows a simplified UI for a client's employees; typically, these are the people who do not otherwise sign in to NetSuite. This can make it easy for them to submit time-off requests or expense reports, for instance.

All these centers require additional (and very limited) licenses to be purchased. For this reason, clients should first consult the NetSuite sales team, which will allow them to understand additional costs and limitations for each type of license before they plan to use them. We'll look at these again in more detail in *Chapter 10*.

In addition to these main products, NetSuite sells the system via a few important editions. Those editions fall into two main categories: **Financials First** and **Starter Edition**. Financials First is what you want if you're a newly established company, looking to nail down your accounting, finances, and CRM with NetSuite at first. You can always add other NetSuite features later on, which the company refers to as the "Stairway." Clients can start on the first level and climb higher as they need more functionality. This option allows a company to start using NetSuite very quickly with the basics – journals and invoices; customers and vendors – but then build on success over time to add additional features as they need. Starter Edition is provided for many different verticals (also known as industries); it is more feature-rich and turns the NetSuite account into an industry-specific system. NetSuite offers Starter Edition for advertising, consumer packaged goods, MFG businesses, non-profits, and much more.

In addition to their chosen Starter Edition, some businesses need extra functionality right from the start. Here are a few of the most popular add-ons or custom solutions some companies choose to start using from day one:

- **SuitePeople:** This is the extension of NetSuite's native human resources features for companies that have more than a few hundred employees, or just have more complex employee management requirements (depending on their locale and industry).
- **SuiteBilling:** Announced a few years ago, **SuiteBilling** is the advanced billing package NetSuite offers to companies who need to support subscriptions or contracts and such. NetSuite has consolidated its billing features under this heading, replacing what used to be known separately as either the Contract Renewals bundle or Subscription Billing. This is typically used by companies in the software vertical.

-
- **Advanced Revenue Management:** Any organization that needs to control its revenue recognition or uses billing schedules can take advantage of this package. This is also typically used by software companies since their product sales are not as simple or direct as physical item companies, and they must adhere to relevant accounting principles. This includes many related features, such as Revenue Allocation.
 - **Advanced Inventory Management:** NetSuite offers what most smaller companies need to keep track of the items they purchase and sell in the basic, starter package, but for companies with multiple locations (warehouses or retail stores) or those that want to offer items for sale with more complicated pricing schemes, the Advanced Inventory Management module is required. Then, there are a few other companies that NetSuite has acquired that offer separate products any company can sign up for. Some of these are also integrated with NetSuite ERP to various degrees.
 - **Oracle CX Cloud:** Marketers and advertisers need advanced features when it comes to getting their message in front of consumers, and this advanced offering from Oracle can help with that process. (This module replaces the previous NetSuite offering, Bronto, which was strictly about marketing via email.)
 - **OpenAir:** For companies that want even more in the realm of service management, NetSuite offers a product they acquired in 2012 called OpenAir. This is a full SRP package in a separate UI that either companies can use in conjunction with NetSuite ERP/CRM or can be sold as a standalone product. It includes timesheets and expense reports, as well as management dashboards and reports.
 - **NetSuite Planning and Budgeting (NSPB):** NSPB allows finance departments to focus on the future by giving them a single place to perform detailed budgets and forecasts, model what-if scenarios, and generate related reports. These users will love the collaborative features that support the development and approval of budgets and forecasts.
 - **NetSuite Analytics Warehouse:** Data warehouses are valuable additions to any business, allowing for the combination of data from many sources. NetSuite's offering in this space runs in the worldwide Oracle Cloud Infrastructure and makes it possible to easily combine data from NetSuite ERP and other modules with data from other sources. Advanced business reporting and analytics are then possible and easily shared with just the right audience across product lines. AI-based automated insights and prebuilt models complement these features to truly streamline the process of reporting for larger companies.
 - **NetSuite SuiteCommerce InStore:** Starting around 2010, Retail Anywhere offered an in-store retail **Point-of-Sale (POS)** system, and NetSuite acquired them in 2013. Since then, Oracle has sold this offering as **NetSuite SuiteCommerce InStore**. This solution integrates a PoS terminal (tablets, cash registers, and so on) with your NetSuite financials, inventory, and order management for a seamless experience. SuiteCommerce is the web store component and SuiteCommerce InStore is the PoS component; they can be used together or separately.

There are many, many more options available to every NetSuite client (far too many to list here). In a typical sales process, NetSuite's salespeople will work to understand a business's industry and specific needs and then recommend just the right list of features and options to be included in each client's package. In all cases, NetSuite tries to recommend what they refer to as the *leading practice* – that is, the set of native and predefined features and components available off the shelf, versus a custom solution for each client. That same approach should be applied to implementations as well, as we'll explain later. Most of the time, those features and options will come directly from NetSuite. However, the system of partners and other third-party application developers for the NetSuite SuiteCloud Platform has grown significantly over the years, so solutions can come from these folks as well.

When we talk about the *NetSuite SuiteCloud Platform*, or sometimes just *the Platform*, we're talking about the customization and integration options designed for the NetSuite product, which allow us to extend the native feature set as we need. NetSuite's designers built features such as custom scripts, workflows, and web services into the product since they knew that a modern application can rarely stand entirely on its own. So, the Platform includes all these additional options, such as **SuiteScript**, **SuiteFlow**, **SuiteTalk**, and **SuiteAnalytics Connect**, which we'll talk about next and then in the implementation context in the last few chapters of this book.

Today, many very important tools and applications are only available from partners via the `suiteApp.com` website or the SuiteApp Marketplace within every NetSuite account's UI. Think of this as the *app store* for NetSuite clients who want to add new functions to their NetSuite accounts without building the customization themselves, which is generally a good idea. These partner solutions are in use already and have been tested by many other clients, which means they offer higher reliability and **Return on Investment (ROI)**.

Here are a few good examples of the most popular SuiteApps today:

- **Avalara AvaTax**: In the US, tax laws vary from state to state, and Avalara makes keeping your sales in line with them all a piece of cake. The AvaTax package can be installed in an account and adds improved tax-related records and fields to **Sales** and other screens where needed.
- **Boomi**: The Boomi service is a large-scale data middleware provider that makes it easier to connect NetSuite to any external service that doesn't already have its own NetSuite connector.
- **eMerchant services**: Provides credit card processing and more for payment services.
- **Expensify**: Provides advanced employee expense management with tight integration into the NetSuite product for any type of business.
- **RF-SMART**: This app supports a growing array of NetSuite's key warehousing, WD, MFG, and retail functionality. It enables customers to fill the functional gap between a barcoding system and a complex WMS.

- **StrongPoint:** For those who maintain one or more NetSuite accounts, with all of the many settings and features and with many people possibly making changes all the time, you need a change management process to keep track of the current configuration and customizations – and that’s what StrongPoint’s tools help with. The StrongPoint spider searches through an account and automatically catalogs all custom objects, scripts, and so on, and notices when they change.

The key with all these SuiteApps is that they are provided by third parties, and so a separate purchase and contract is needed in each case. As well as the SuiteApp website, you can check out the relatively new SuiteApp Marketplace within the NetSuite UI. Reach out to those other companies for more information. For more complex SuiteApp solutions, NetSuite clients will need a separate services contract and support agreement with the app’s provider. Keep this in mind when you’re scheduling an implementation!

It’s important to say here that as a NetSuite consultant, one key aspect of your job is to make sure your clients know about all the viable options available for solving their problems, including the SuiteApps out in the marketplace. Depending on how your business is set up, you may also recommend some solutions, but ultimately, the client has to make the final decision on which to contract with, for the good of their business. Work with your company’s people and the client’s management to guide them to the right choice – the one that will serve them best in the long run.

Now that you know a bit more about the NetSuite product and the options we have for enhancing its features, I’ll explain a little about what we should do when those options are not enough to meet a client’s full requirements. This is where customizations come into play.

The NetSuite SuiteCloud Platform

When none of the earlier options will satisfy a particular client requirement and they are adamant that their business cannot function without it, then customization built with the SuiteCloud Platform will be needed. We usually refer to these situations as *gaps* in the client’s requirements since they are not covered by native NetSuite features, including simple custom configurations or any existing partner solutions.

In these cases, you can help the client define their requirements and then design a customization to satisfy them. Depending on the complexity of the gap, this can mean a short delay while the custom solution is created, or it can sometimes become the largest chunk of work being completed by the implementation team. We’ll discuss these types of customizations in *Chapter 18*, but for now, just note that you should always work with the client to find acceptable workarounds to avoid having to create a custom solution on the NetSuite platform. I spend most of my time at NetSuite designing and building this kind of customization as I work with my clients, but I only do this after exhausting the easier/faster options. Nobody wants to delay the client from going live, so avoiding customizations whenever you can is very important.

Again, though, sometimes, you just have to customize NetSuite. That’s why it’s a very good thing that the system’s architects have given us the tools we have today, which we discuss in the following subsections.

SuiteBuilder

This feature includes custom records and fields, subtabs, and sublists. With these, we can add a custom table for things such as warranties or shipment packages, built just right to suit the client's needs.

This also includes things such as custom segments for when departments, classes, and locations are not enough. We can use custom transactions for those rare times when the native transactions don't allow enough flexibility or a client has some other special requirement.

SuiteScript

Scripts are automations or customizations that allow almost any special processing to occur, when you need something more than NetSuite offers out of the box. Many types of SuiteScripts are available, including the following:

- **Client side:** In the browser, this is used whenever a page is first loaded, a field value changes, a line is added to a list, and so on
- **Server side on a user event:** These are a little more limited in their triggering moment, including before the page loads (`beforeLoad`), after the user clicks **Save** but before data is written to the database (`beforeSubmit`), and after data is committed to NetSuite (`afterSubmit`)
- **Suitelets and RESTlets:** These scripts allow you to create custom UI pages as needed or make microservices that listen for requests coming into NetSuite from other external sources
- **Scheduled and Map/Reduce:** These background processes run on a predefined schedule, such as once per day or every two hours, for those times when you need some data to be updated but it's not urgent
- **Miscellaneous options:** These include scripts for creating custom dashboard **portlets** and scripts that affect the outcome of a mass update, among others

One common reason for creating a script is when a client wants to limit the values possible in native fields, for instance. In those cases, NetSuite doesn't let us use the configuration screens to directly control native fields the way it does custom fields, so we typically write a script to change the behavior of the field instead. Other examples of a commonly used script type are GL Lines Plugin scripts. These allow us to customize the native GL impact of a variety of transactions when the client's accounting rules are different from NetSuite's.

SuiteFlow

This is NetSuite's custom workflow, configured with a point-and-click interface in the same NetSuite UI we use every day, allowing more control over things such as approval processes, email notifications, and simple validation/error handling.

For instance, if a client wants a warning to be displayed whenever a quote is saved without two fields being populated, a workflow can do that easily.

Most intermediate to advanced NetSuite users can learn to create these when needed, so they're NetSuite's *no-code* alternative to writing scripts.

But just note that, for now, if you don't know what you're doing, it's also easy to create a workflow that harms performance or, worse, causes a NetSuite screen to misbehave. For all these SuiteCloud tools, getting trained first is a really good idea to avoid this kind of trouble. We will cover SuiteFlows in more detail in *Chapter 18*.

SuiteTalk (SOAP and REST)

NetSuite introduced SOAP web services many years ago, and many third-party connectors still use it. **SuiteTalk** is essentially a service within the NetSuite cloud that is always listening for web-based (programmatic) requests to be sent to it, and it then reacts to those requests in a predefined way.

For instance, if your third-party web store needs to tell NetSuite that an order was just placed, you could develop the SOAP web calls to do this, and NetSuite would react by creating the matching order in the NetSuite database.

SuiteTalk integrations are created outside of NetSuite, so you need a server of some sort for them to run on. NetSuite doesn't specify the details around how you do that, so you have a world of options in that regard.

More recently, NetSuite introduced the SuiteTalk REST API, which works a little differently from the SOAP interface but serves the same purpose. Whereas SOAP messages are sent in XML format, REST uses JSON, so this might be familiar to more developers today.

SuiteAnalytics Connect

The **SuiteAnalytics** set of features includes all of the native reports, datasets, and workbooks. Those things are all great if you're already working in the NetSuite UI, but that's not always true for all the employees of a typical NetSuite client.

SuiteAnalytics Connect is an ODBC/JDBC/.NET interface you can use to extract data from a NetSuite account to any external database or data warehouse. This is a good option for when you want to pull data out of NetSuite regularly and use it in another system or database.

A common example would be a client who needs to export all inventory and sales order data from a NetSuite instance into a data warehouse (such as NetSuite Analytics Warehouse) so that they can slice and dice the data for custom reports.

All of the powerful options available via the NetSuite SuiteCloud Platform enable businesses to make the system work just the way they need it. This is a major differentiator for the product and improves user efficiency as well.

NetSuite's mobile apps

In addition to the standard NetSuite UI, which works best on non-mobile devices, NetSuite also offers a set of handy apps for Apple and Android devices to meet the needs of users who are away from a desktop computer:

- **NetSuite Mobile:** The flagship app for any NetSuite user, this app allows users to enter expenses and time entries, approve transactions, log activities and events, receive reminders, and much more.
- **Manufacturing Mobile:** This new offering allows manufacturers (with or without NetSuite WMS) to gain easier access to MFG transactions (work orders, assembly builds, and so on) from a mobile-friendly interface. This is not an app you get from an app store, however; it's a bundle you add to a NetSuite account, which then creates a new URL that your mobile users can access in their browsers.

Here's an example of the dashboard in the current NetSuite Mobile application for Android and iPhone:

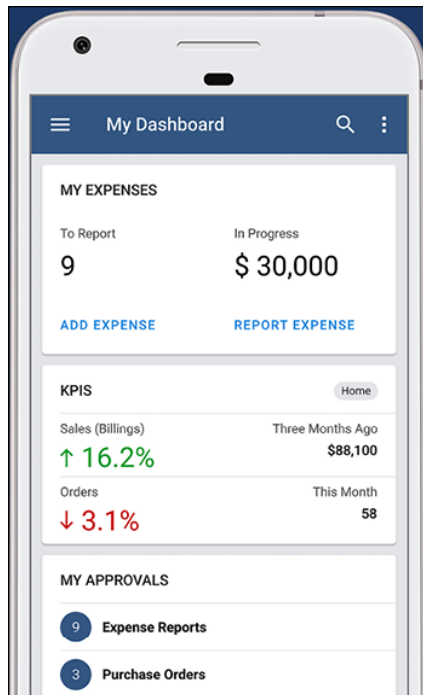


Figure 1.2: Example of the NetSuite Mobile dashboard page

These mobile applications make it easy to access NetSuite when you're away from your desktop or laptop PC. They don't have the same full suite of features as on PC but they can still help a lot, particularly in places such as your client's warehouse or other operation centers.

Summary

In this chapter, you learned how NetSuite is set up and works, how the company is organized, and what you can do with the NetSuite product.

Armed with this knowledge, we can explore the first steps we must take to implement clients on NetSuite – the methodology we will use and how we can apply it in the real world.

In the next chapter, we will move on to the topic of choosing an implementation methodology. This is critical to the success of every NetSuite implementation, and with this in mind, you can communicate the details of your process methodology to your clients.

Self-assessment

I'm going to include this section at the end of every chapter. This will allow you to think about the topics that were covered and see how they apply to your work:

1. When does a company need to move up from the base tier in NetSuite to a more premium version? Have you worked with a client who has needed this?
2. If you create something custom in a sandbox, what's the easiest way to move it into the Production account?
3. If you're confused about all of the available modules that can be used along with your NetSuite ERP account, which team at NetSuite is best suited to helping sort things out?
4. What is the difference between SuiteCommerce Advanced and SuiteCommerce InStore? What are the benefits of each product, and what kind of company should purchase each option?
5. A client wants to allow their users to enter time and expenses while they're working at client sites, on projects. Which NetSuite features can you recommend they explore?

Learn more on Discord

To join the Discord community for this book (<https://packt.link/UytqI>) – where you can share feedback, ask questions to the author, and learn about new releases – follow the QR code:



Selecting and Applying an Implementation Methodology

In the previous chapter, we covered NetSuite, its many features and products, and the ways you can extend it. Keep all of that in mind as we start applying project methodologies since you have to know the product well to implement it successfully.

Understanding implementation methods is a good first step on the path to performing implementations yourself. In this chapter, you'll understand why having a clearly defined implementation methodology is so important to both finishing a project successfully and completing the work on time.

In this chapter, we're going to cover the following main topics:

- What is a methodology and why do we need one?
- Selecting from the different approaches
- The Waterfall method
- A more Agile method
- Adapting the methodology to each client

By the end of this chapter, you should have a solid idea of the method you will use to implement your clients on NetSuite.

What is a methodology and why do we need one?

We need to talk about what implementing NetSuite means and how we do it. If someone knows their business and they're pretty savvy with software, they might think they can get their business or non-profit organization running on NetSuite without help from anyone outside their organization. That might be true if the company only has a few people and they've got every single business process those people deal with well mapped out. Also, it helps if they're all in the same location, work on the same shift, have simple taxes (or ones that are not a concern), and so on. However, for most companies starting with NetSuite, for many reasons, getting help with the implementation process makes a lot of sense.

We'll dive into that idea in this chapter, explaining how and why help might be needed, as well as what you need to understand before you take on this challenge yourself. My goal is to pass along as many tips and tricks along the way so that you'll save time (and money!) and do everything you can to ensure the success of your projects. In the next chapter, we'll break down how to plan the entire project, but first, we need to talk about *methodologies*.

In the software consulting world, we always want to show potential clients that we have experience with whatever our subject matter is, or even better, that we're experts in it. One way we do that is by having a clear, concise way of communicating how we will approach their business when completing their project. That is what we mean by methodology in this chapter: the process we follow, typically with all our clients, which has been proven over time to result in the fastest and most successful go-lives.

In NetSuite terms, the methodology we adopt serves a few purposes:

- **Understanding their business:** This helps us convince potential clients that we know what we're doing, we've done it before, and we can do it for them too
- **Streamlining the project's day-to-day work:** Having a preset methodology also serves the purpose of making it easier to communicate the status and next steps since the client will know from day one what all the steps in the process are
- **Avoiding delays and surprises:** If we tell a new client up front that we need to, for instance, gather requirements before we start building anything in NetSuite, then they have a clearer expectation of what each day will be like as we proceed through the project

There are quite a few software consulting methodologies out there, each using its own terms and steps for the process, but most of them share the following steps:

1. Kick off the project and get to know the client and their requirements.
2. Design the right solution to meet the client's needs.
3. Spend some time configuring things and building that solution.
4. Test everything from end to end to make sure everything works together correctly.
5. Train everybody, get all the data set up, and then go live in NetSuite!

This same kind of flow is used for most software, but a NetSuite implementation is a special case and can be more complicated than implementing something like a CAD program or tax package. This is because it will be used by everyone in the company in some regard.

Let's take a look at the most common approaches that are taken in the field to address this complexity, and each method's strengths and weaknesses.

Selecting from the different approaches

If you talk to a group of NetSuite partners and consulting organizations, you will find they each advertise their own somewhat unique methodology for getting clients up and running on NetSuite. They will each assure you that their approach is the one you need and that it nearly guarantees success, but please take claims such as these with a grain of salt. Most of these companies are not reinventing the wheel and they are, in fact, following one of the two most popular methodologies in the 21st century.

The **Waterfall method** is a one-pass system for going from requirements gathering to design to building, testing, and delivery, and this was the process explained in the preceding section. This approach has been around for a long time in the software world and generally speaking, it's usually successful. It works best for consulting organizations that are interested in selling a predefined, contracted service, without a lot of room for changes. This is great for consultants because they know what they're committing to from the beginning and what they will be paid for. This doesn't always work out well for every client, though. If they are not 100% clear on what their requirements are, or if they need a more dynamic, flexible approach, then problems could arise with the Waterfall method.

More recently, some companies have chosen to implement NetSuite using an **Agile approach**. That term is very loaded and will mean something different to most people you talk to. Without going into a lot of history, the software development world largely started shifting to an Agile approach sometime in the late 1990s/early 2000s, and for good reason. When you're building a product, one that will be around for a long time, you need to continually add features and make users happy over time. You need a way to gauge what your users want from the product, and you must plan to add the features they need while working with them in manageable time frames. This kind of approach can also work well in a NetSuite implementation sense, but it is important to understand the advantages and disadvantages of each method.

Here is a flowchart explaining how the methodologies differ in terms of providing value over time, compared to the risk you incur in using each method:

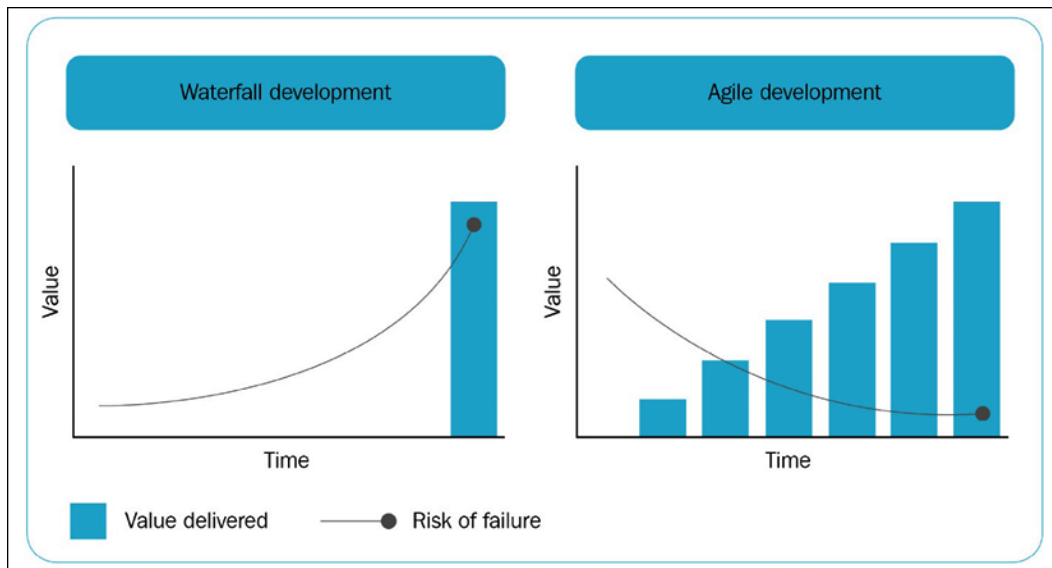


Figure 2.1: Time and value delivered for Waterfall versus Agile methodologies

I should mention here that if you do some online research, you'll find an endless list of project methodologies used by NetSuite, partners, and other experts. Most will fall into either the Waterfall or Agile camps though, in most respects.

The Waterfall method

As explained earlier, the Waterfall method looks like a trip on a railroad. When you board a train, you know what your destination is and how long it's going to take to get there. You paid for your ticket upfront and expect to receive all of the advertised services while you're traveling. This kind of project should be instantly understandable to most people and that's just one of many reasons why most NetSuite Professional Services implementations follow this plan – initially.

Projects implementing the Waterfall method are often tied to **fixed-bid** contracts. These contracts are frequently referred to as a **Statement of Work (SOW)** for consulting services. These tell the client exactly what the consulting company will do within the scope of the contract. They are usually written to make it clear that only the work spelled out in the SOW will be performed; anything else needed by the company will require a separate or additional contract, and that will incur additional costs.

Here are a few things you might hear from clients in your *discovery* calls, which should lead you to choose a Waterfall method for their implementation.

“We have a very limited budget for this implementation, so we cannot afford to have any surprises or pay for additional costs once we start.”

- With Waterfall, the parties usually sign a contract at the start that states exactly what is and is not *in scope* for the services to be provided.
- This can give the client a good sense of confidence in the amount they will pay for that service.

“We know our business well and we know what we want to achieve with NetSuite.”

- This makes Waterfall the right choice since it assumes clear requirements can be stated very early in the project – and the client will stick to just those throughout the entire set of work.

“We need to be up and running on NetSuite in the shortest time possible.”

- With Waterfall, once a plan is put together, everyone can see the entire journey from the start and can predict when it will be completed (assuming all goes well).
- This approach usually means less back and forth over what should be done and how to do it, and that usually translates into spending less time getting the client’s people and everything in NetSuite ready for day-to-day use.

When we are delivering a fixed-bid contract and using the Waterfall approach, we talk to the client about their requirements. Requirements are usually stated in terms such as the following.

“ABC Company requires that users completing purchase orders in NetSuite be capable of tracking estimated manufacturing and drop ship dates for the transaction and that these dates be calculated for them automatically, so they can efficiently communicate with customers waiting for orders.”

The Waterfall methodology can be a good fit for companies thinking along these lines, but even they are not always aware of complexities that can lead to less successful results for a Waterfall project. Here are some ways that Waterfall projects fail if not properly managed:

- They do not allow for much flexibility or many changes once started.
- In most fixed-bid contracts, we list the explicit services – in some cases including very specific details regarding what’s in scope – and we usually have a paragraph that says that any additions or changes are not covered by the same contract.
- During the implementation, if the client changes their mind about what they need – say, if a wholesale distribution client suddenly realizes that they need manufacturing features, and so needs help implementing something that was not spelled out in their contract – that is usually handled as a *change order*. That’s a new, separate contract, spelling out what’s being added to the project, and the additional cost involved.
- Changes can easily delay the schedule in a Waterfall/fixed-bid project, so everyone involved should be on the lookout for potential issues and changes of mind that can lead to them.

- Even when the business has to change, the fixed-bid contract does not.
- Depending on a lot of factors, Waterfall NetSuite implementation projects can take between three months and a few years (in extreme cases) to complete.

During the project, a business still runs its day-to-day operations and changes will pretty much always occur within the business, moving us away from what was initially considered the go-live plan.

For instance, a business might have one subsidiary today, but then find itself in a good position to acquire another company (or they could be acquired) during the implementation. Or they might open a new warehouse or offer an entirely new product line during the implementation. These kinds of things usually lead to new requirements that were not spelled out in the SOW.

- Some organizations simply cannot do a good job of stating their requirements.

You would be forgiven for thinking that anyone who has worked for – or led – a company for a few years would understand all there is to know about how they perform their accounting, ordering, purchasing, and so on. However, that is not always the case.

Most frequently, in my experience, the corner cases or exceptions to the rule are forgotten at first during a project.

The client may have told you about a requirement upfront without explaining it fully and so leave out key details that will affect the design, build, and test schedules in a material way.

These kinds of surprises are frequently the source of problems within a fixed-bid/Waterfall NetSuite implementation project.

So, how do we deal with problems like this? One solution is the Agile method.

The Agile method

The Agile software development approach came to life in the late 1990s, when software developers started to realize that the inflexibility of Waterfall projects was killing their productivity, delivery schedules, and bottom line. We'll explore this idea here and talk about why this works so well for product development teams and why this isn't nearly as good of a fit for fixed-bid software projects, which are typically delivered by professional consultants.

With Agile software development, we can adapt to changes – in terms of requirements, timelines, and budgets – more easily than with a Waterfall process. Product developers, who generally build one product over a longer time frame, love this idea because as stakeholders bring new requirements to them, they can weigh up how to fit those into the existing architecture in a way that stands a chance at being successful. Many of these teams have adopted some form of Agile in the last 20 or so years, most very successfully. The most common methodologies they talk about are **Scrum** and **Kanban** (which is very much like Scrum but without timed *sprints*).

With the Scrum framework, a Scrum Master leads the product team through a process whereby some of the initial requirements are gathered (in the form of *user stories*), and then the development work is defined and prioritized in small, attainable chunks known as *sprints*. Once the first set of stories has been set and their importance has been decided on, development and testing can start. Every so often, typically every two weeks, they reevaluate the work that's been done up to that point, reprioritize the remaining work, and push on. That two-week period is known as a sprint. The emphasis here is placed on maintaining that level of flexibility over time, deciding what to work on next based on the current conditions at the end of each sprint.

This is how we usually want to see a typical sprint happen but the timing and other details can vary a lot, depending on the work to be performed:

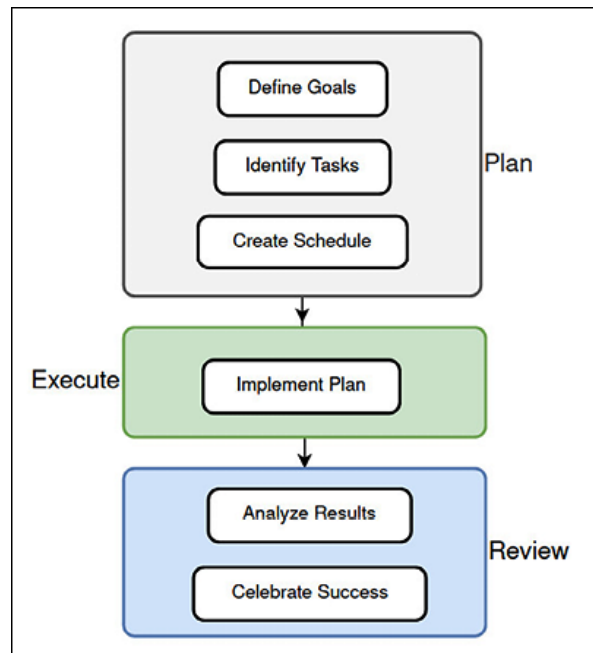


Figure 2.2: The steps in a typical Agile project sprint

If we apply that thinking to a NetSuite implementation, we might start the project with basic configuration and training for new users, but then, depending on the things that come to light in the next few days or weeks, we might shift our focus to new problems that arise to make sure they are addressed according to their urgency. As a client reports issues or brings new requirements to light, we would add tasks to a work queue, and then we would make sure they were completed in whatever the project's steering committee felt was the right order.

Problems can arise, however, if we stick with this purely Agile approach for the duration of the project. To be blunt, clients can misdirect the implementation team by bringing up new challenges as they occur to them (for instance), in such large numbers that we might never actually finish the work everyone agreed on at the start. For example, at the beginning of an implementation project, a client might say that getting their order-to-cash and e-commerce sales configured right are their two highest priorities, so the project team might plan to spend most of their time working on those things. However, a couple of weeks in, if a purchasing issue comes to light that looks like it might block all the work on the sales side, the team could prioritize that over finishing the setup of sales orders. You can see how, if that's not carefully managed, the work on the sales side might be seen as being late or incomplete soon thereafter, and that's where a project can start to fall apart.

Most experienced **Project Managers (PMs)** know that to keep the client's trust – and to get them *over the finish line* on time – proper project management routines must always be followed. In this scenario, that would mean the PM should always work with the client's project team to confirm any shifts in priorities, as well as to clarify the effects moving people from one task to another will have on the overall project timeline. Generally speaking, it's this specific concern – the possibility of missing already-agreed-to deliverables or completing them late – that should make you wary of taking a purely Agile approach.

Instead, I generally prefer the tried-and-true Waterfall approach since we know what the work will consist of at the start; anything new that arises is a change order or separate contract, or is called out as happening after the current work called for in the delivery schedule.

Having said that, I've worked on a few projects where the client insisted on an Agile approach, and in those cases, the implementation team came together and agreed to work the Agile way to keep the client happy. However, in these cases, that would mean avoiding talking about high-level requirements and instead talking to the client in terms of user stories.

I gave an example of a requirement earlier, but if we wanted to turn that into a user story, we would need to reword and break it up a bit. The most common *template* for a user story reads like this.

"As a [user role], I need to [do something], so that I can [achieve some specific benefit]."

Our goal with a statement such as this is to define something small enough to work on right away and complete very soon, which can then be tested easily to confirm we did it right. So, to adapt the preceding purchasing requirement into user stories, we might say the following:

- **User story 1:** As a purchasing agent at ABC Company, I need to see an estimated manufacturing and drop ship date on purchase orders so that I can tell my coworkers when they should expect the products to be available for sale
- **User story 2:** As a purchasing agent at ABC Company, I need the estimated manufacturing date to be automatically populated via data imports from my suppliers so users don't have to enter the date manually

- **User story 3:** As a purchasing agent at ABC Company, I need the estimated drop ship date to be automatically populated as the estimated manufacturing date plus 14 days so that users don't have to enter the date manually

The difference has to do both with the level of specificity and with how many details we include in each statement. We write *requirements* to be as specific as needed to ensure that we understand the client's needs, but not so specific that we can't make minor adjustments later. This is important since with requirements, once we agree to these terms, that's pretty much exactly what the client will get, and in some cases, months can pass between when the requirements are gathered and when the final product is delivered.

With user stories, though, since we're trying to define smaller chunks of work within the larger goal we're trying to achieve, they should be more specific, only cover part of the overall requirement, and be easily testable by any knowledgeable user once delivered. User stories can describe both things we do using native NetSuite features and things that require some kind of customization.

Once we have a set of user stories written down, those go into a work queue, a priority is assigned to them, and the work begins. If anyone thinks of something new that also has to be done, a new and separate user story is written and it gets added to the queue, usually, but not always, at the end. Some clients love this because it means they don't have to spend nearly as long at the beginning of the project explaining every last detail of their business to us before the work can begin. That type of client usually appreciates the idea that they can change their mind – on pretty much anything – and we'll just keep working with them for as long as it takes to make them happy.

If you're paying attention here, you can probably already see the big downside of this approach: the cost. Agile projects must start with a **time and materials** contract, which either will say that the implementation team will continue working on whatever the client directs them to until all of the client requests have been addressed, or can have a list of high-level topics defined instead. Those would act as a limiter on the scope of the work the team will complete within that contract. The client then agrees to pay for that work by the hour. When we enter arrangements like this, we usually estimate the total work upfront and put a cap on the total number of hours or dollars that can be billed under that contract, just because almost no client wants to sign a limitless agreement.

If the work is going well, as that maximum is reached later on, then a new contract can be defined, saying more work is needed or the original contract can be extended. However, that's also still a good time to stop and evaluate things very carefully to make sure that the client still has a budget for more work and the stomach to delay their go-live date, for instance.

Going down this Agile path is risky, for sure. The leaders of the implementation team must know the client's management team well so that they can read their responses to the key questions posed early on, and also know whether they can afford to go with this more open-ended method.

Many companies think they want this early on but then change their minds when they see the total of many months' bills piling up and still no clear go-live date in the plan. As consultants leading an implementation, we must avoid this result at all costs.

Case study – Machine parts distributor

To provide a real-world example of a project where having a chosen methodology and following it really saved us, let me tell you about a machine parts distributor I worked with a few years ago. When the project started, we had a fixed-bids contract to deliver one customization along with the normal implementation. We created and delivered the customization, and then allowed the client to test it. They liked it well enough, but wanted more customizations at that stage. We talked this over with them and were able to convince them to hold off on automating more of their processes until after the go-live. Not many clients are as amenable to this discussion as these folks were, but it's still always important to try to deliver the project we started with, as promised, whenever we can.

If you want to learn more about the Agile methodology, consider reading *Managing Software Requirements the Agile Way* by Fred Heath (ISBN: 1800206461).

So, what's the right approach, then? Well, that depends on many things.

Adapting the methodology to each client

Both Waterfall and Agile have advantages and disadvantages that can result in a successful project or something you'll be less likely to want to tell everyone about. In my opinion, a combination seems to work best for most organizations implementing NetSuite.

Starting with Waterfall, applying Agile when necessary

Most companies will know what they need to achieve in NetSuite before they can start running their business on the product/platform. If your sales team is savvy, they can gather most of that information from them during the sales cycle and get a fixed-bid contract written up describing exactly what is needed from the implementation team to support their goals. This usually makes the most sense to all parties, since the costs and delivery schedule can be well-planned based on a well-written, fixed-bid contract. Clients appreciate this clarity since most see the NetSuite implementation as a big expense and a time-consuming affair for their employees (which it is!).

Then, once the project's activities begin in earnest, the implementing PM and the client's PM can work together to identify risks, changes in requirements, and so on, and deal with them in the best way at that moment. Depending on your business model, this might mean a new contract is drafted or changes might already be included somehow. If a new contract is needed, that can either be another fixed-bid contract, if everyone agrees that the changes or new requirements are clearly understood and the client wants to keep the costs to a minimum, or it can be time and materials, generally covering any new work that must be supported at that time. As described earlier, this latter option not only allows us to still focus on delivering just what's in the original fixed-bid contract but also lets us add new work as it comes up. Your company might choose to handle these contracts differently, of course, but many service groups follow something like this; that is, a hybrid methodology.

For instance, following the completion and delivery of the initial fixed-bid contract's work, your team and client might agree to complete additional work on a time and materials basis. In that case, the workflow within that time and materials contract's scope might look something like this:

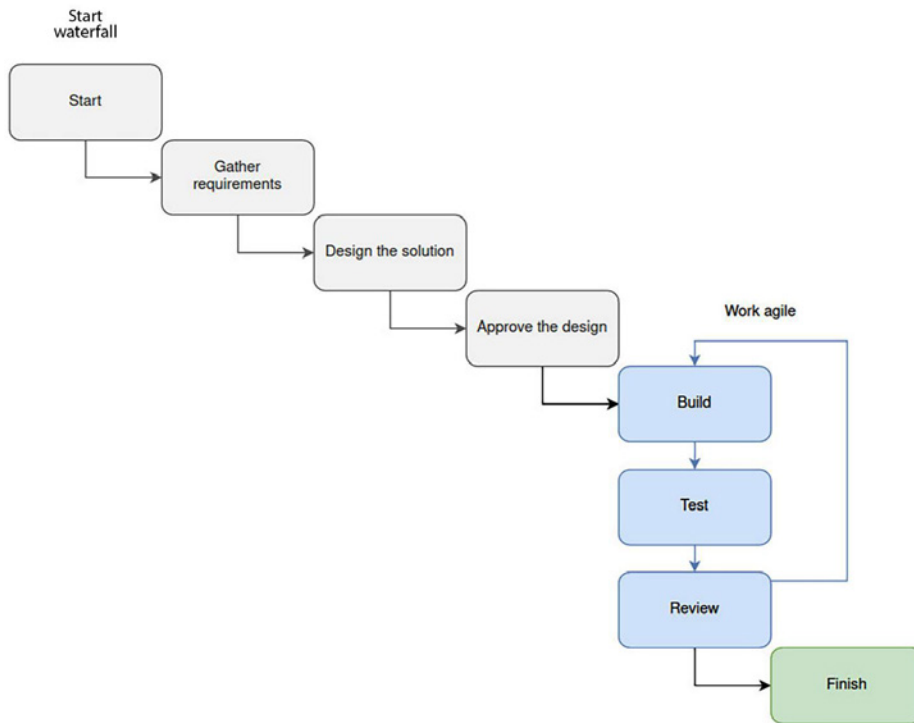


Figure 2.3: Workflow for an Agile approach after the initial work is delivered

In the preceding workflow, we would want to start by understanding what new work is needed – consulting, customizations, help with reports, and so on – and then we'd iterate over that work and keep working until the client was satisfied that they had everything they needed. Depending on the client's mindset, this can all happen before they go live, or this additional work can start after they go live with the initial implementation. In that case, we usually refer to the add-on work as *Phase 2*, *Phase 3*, and so on.

The advantages of this are twofold: we start from a place of clarity and direction, but we allow changes to occur and handle those as flexibly as we can. We never lose sight of our original goal, and we ensure that even if we don't get the client to a *live* state on NetSuite, we can at least deliver everything mentioned in the first contract.

Then, as we add new requirements or user stories, they are completed in a separate timeline. The client has control over when we stop building new things and tweaking how NetSuite will work for them at any point, which means they can go live as soon as they are ready.

Summary

In this chapter, you learned what a project implementation methodology is all about, why we should have one in mind as we start to plan any NetSuite implementation, and what makes each method better or worse than the others.

Hopefully, this has given you a sense of why we need a methodology for our implementations, and how they're typically defined. With an understanding of the differences between a Waterfall and an Agile approach, I hope you have a good sense of when you might want to use each method.

In the next chapter, we will learn how to plan a specific project, including all the details the plan should include and how each person involved in the project plays an important role. Just keep in mind that you will want to apply your chosen methodology to the specifics of your project plan and see how it all plays out over time.

Self-assessment

Here are some things you can think about on your own, related to this chapter's contents:

1. What would your projects look like if you didn't start with a methodology in mind? Do you have a better idea than those presented here for a project method you will follow?
2. Since a Waterfall project starts from a defined list of deliverables, how will you talk to your clients about changes they request in the middle of the project?
3. With my proposed hybrid Waterfall/Agile approach, when is the right time to explain the approach to a client? Upfront, as part of selling your service to them, or only when they start to ask for changes during the implementation?
4. How does your business currently handle adapting to changes the client asks for? Are you in the strict the-contract-is-all-you-get camp or are you one of the all-you-can-eat type of service providers? Which is the best approach, in your opinion?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search this book by name.

Note: Keep your purchase invoice ready before you start.



Creating a Project Plan

In the previous chapter, we learned how methodologies affect the outcome of any NetSuite implementation and how to evaluate your approach using one. Every time you start a new NetSuite implementation, you'll need to develop a project plan, spelling out the steps in the process – that is, showing your methodology in action but also adding whatever makes that project unique based on the client's business. Here, we will cover some tips for making each implementation project a success.

In this chapter, we will learn about the following:

- What goes into a project plan?
- Conference room pilots, walk-throughs, and testing
- Applying your methodology to the project plan
- Living with the plan as changes occur

With this understanding, you should have a clearer idea of how a project plan can help you kick off your implementation projects on the right foot, and you should complete them successfully if you follow this plan throughout the project. Using the ideas here, you will learn how to test whether your team is on track and how to fix things when the client or anything else changes the project's trajectory.

What goes into a project plan?

Now, we need to see what delivering an implementation project requires in more detail. Like any project, we need to have a goal or set of goals in mind, so we know whether we're moving in the right direction. Those are going to vary from one organization's implementation to the next since sometimes speed is of the essence and other times, getting everything set up just right is more important, even if it takes longer.

Once we have goals, we can lay out the path for achieving them, and make sure all parties involved know what will be expected of them. In the software delivery world, setting expectations correctly is probably the first and the most important thing we do. Many of the problems I've seen happen in large and small projects can be traced back to someone having the wrong idea about what they were supposed to do or what others would do for them. Getting through a NetSuite implementation to the day when the organization decides it's ready to conduct its business in NetSuite is a big achievement, but it typically takes a good deal of work to get there.

Any NetSuite implementation project plan – whether tiny or huge – should include the following elements to be successful:

- A version number or a date indicating the time the plan was most recently updated. We'll keep this current as revisions are made since we know that is almost guaranteed to happen.
- A section at the top explaining the goals we're setting out to achieve. We will see in the next section how to decide on the right goals for any project.
- A list of tasks to complete, including dates and durations for each of them. This is sometimes tough to establish realistically but do your best anyway since this is how we plan when the client might be able to go live.

People's names are associated with each assignment. In the beginning, this might not be a specific name but a team of people instead, such as the *accounting department* or *functional consultants*, but more detail now will help drive the project to succeed in the long run.

Next, we'll focus on how setting goals is a key element in the planning process and talk about how you set and apply them throughout a project's life cycle.

Setting project goals

When we first start to create a project plan, there are many unknown things and many uncertainties. That's OK; we still take the time to document what we're sure of and identify those things we're not clear about, as this will help us work on them later.

For instance, sometimes the go-live date is the one thing we know we have to make happen. Some clients might decide to transition to NetSuite when their legacy **Enterprise Resource Planning (ERP)** software won't be supported any longer. Or, maybe they could continue to use the older software, but only if they sign on for another few years. In these cases, the client will be highly motivated to stop using the legacy software and start using NetSuite ASAP. When the timing of the project is paramount, we frequently have to remind the client that compromises in product features may have to occur. They can't say they want to go live in 90 days and also say that a complex custom solution must be created and ready for use on day 1, for instance.

For other organizations, the go-live date may not be the driving factor, but getting certain functionality working just right is. For these groups, a feature such as return and warranty handling, or the ability to control their pricing in very specific ways, may trump everything else in their minds. In these cases, we can target a later go-live date and focus on getting their NetSuite account set up to work as they need first. Just as before, when we're focused on building functionality that NetSuite doesn't include out of the box, we may need to remind the client that these efforts add to the delivery schedule, and so a later go-live date is usually the result.

When deciding on the project's goals, it's important that they are measurable and realistically achievable. It's not helpful setting a go-live date that you know your team can't handle, or saying you'll build customizations that your team has no experience with. It's fine to have a little uncertainty about these kinds of things, but just remember that once the goals are set, you will need to map out the course – show the client how you will achieve the business' improved state in the agreed-upon time frame.

Here are some fictional examples of project goals. You can use them to model your own once you understand your client's specific requirements. We'll cover that process in greater detail in the next chapter:

- Company X wants to completely replace their current ERP system with NetSuite, with **general ledger (GL)** reporting and purchase accounting functions being the most important features
- For Company Y to succeed in the marketplace, its sales force management and monitoring must become more agile and faster, and sales reps must be able to use it while out in the field
- ABC Inc. needs to modernize and streamline its warehouse operations and fulfillment processes to increase brick-and-mortar store sales in order to compete

Most companies will have more than one goal, and you can make a shortlist of goals for any client, since they may be looking to make improvements in multiple departments/metrics/service groups, and so on. Just remember that at the end of the project, you should be able to show them how you did exactly what you planned to do, so make sure these goals are achievable and there aren't so many that you get all tied up trying to complete all of those tasks. Knowing when a project is heading toward too much complexity or might have too many goals takes practice and experience, so, like everything else, hiring knowledgeable, experienced **project managers (PMs)** helps prevent this problem.

Here are some examples of poorly written or impossible-to-achieve goals you should avoid writing into your plans:

- **123 Inc. wants to triple sales of their line of children's toys within 1 year:** That's specific and therefore measurable, but also not something NetSuite can specifically help a company achieve
- **The Klein Furniture company wants to become the largest distributor of home furnishings along the Eastern US seaboard:** This is not specific enough as it doesn't say anything about how that might happen, and it also depends on the size and success of other companies, which is something we have no control over
- **XYZ Inc. will go live within 30 days in order to avoid renewing their current ERP system:** That time span is usually too short, even for the best NetSuite implementation team

As a last word on project goals, remember what I said in the previous chapter – changes are bound to happen, and when they do, it's easy to forget your initial goals. To some extent, you will learn to roll with the client's changing focus from day to day, but you have to manage the change, communicate frequently, and get approvals as needed for any new directions. But remember, don't let the project's time run out with you having nothing to show for it. You must show the client that, even though problems arose, priorities shifted, and so on, you finished the work that was agreed on at the start. Ideally, you will take the client live on NetSuite when you get there, but in some rare cases, the initially agreed-on work may be all you have.

If enough *blocking* problems occur, friction develops between the implementation team and the client, and delays and extra costs are sneaking in, you always want to be able to explain why you're at the stage you are and how you stuck to the plan until the client told you to change direction. You do not want to end up in a place where the scheduled go-live date arrives and the delivery team has to explain why they're not done on time or why they didn't work on the right tasks.

Defining the tasks in the project plan

Next, the implementation team's PM should lay out the steps in the plan and assign each one a start date, duration, and resource name. Because every client is unique and has their own requirements, knowing all the steps you might need to include in any one NetSuite implementation isn't in scope for this book, but we can certainly identify the high-level tasks that should almost always be in scope.

As you read this list, just know that there are many NetSuite features not covered here. You will probably need to add more sections and more tasks to any single project, and in some cases, you may be able to remove certain tasks as well. Some companies only want to use NetSuite as a GL accounting system, for instance, so in those cases, the project team shouldn't waste any time setting up items, sales, or purchasing. Other projects might be almost entirely driven by data integrated into NetSuite, so user training and procedures are minimized.

These are generally the main sections we include in most NetSuite implementation plans:

- Requirements gathering
- Initial general account setup
- Bundle installation
- Financials, chart of accounts, and so on
- Data migration planning
- Record-to-report setup
- Design-to-build setup
- Call-to-resolution setup

- Procure-to-pay setup
- Order-to-cash setup
- Setup for other features as needed
- Data migration testing
- Walk throughs or conference room pilots
- Final cut-over/data migration imports
- Go live!

There can be many more important sections in a plan, especially if other systems (such as payroll, an e-commerce web store, or a warehouse management system) are being stood up, or if integration and customization tasks are required. Also, in larger implementations, we frequently need to plan for go-live in multiple phases, over a period of weeks or months, with a clear statement of which features the client will start using with each go-live. So, take this as a starting place and make sure you adapt your plan to your client's needs every time.

In this book, we will cover many of the NetSuite feature-specific areas in the preceding list in detail, in *Chapters 13, 14, and 15*. However, for now, let's dig into what those walk throughs, conference room pilots, and user acceptance testing look like.

Conference room pilots, walk throughs, and testing

Once the client's requirements have been gathered and their NetSuite account has been configured to match the business' needs, it's time to show it off to the business users, take note of any new concerns they raise, and eventually, get their sign-off that everything is ready for them. We don't want to wait for a conference room pilot or walk-through to start checking in with the client on these things. So, let's break down what this process usually looks like.

A good implementation should always start from the knowledge of what works best for most companies. We will call them *best practices* here, but you can call them whatever you prefer; the aim is to start every implementation with these practices as a starting point.

These best practices should also align well with the native features and settings in NetSuite. It's no good telling a client they should set up their accounting in a way that NetSuite doesn't support at all.

If you instead start by asking the client, "How do you want NetSuite feature *X* to work?", you are already in a bad place, for two reasons:

- Most NetSuite clients expect their consultants to already know how a business or not-for-profit in their industry operates, for most topics
- Asking that question opens the door to a lot of conversations about customizations, and we always want to steer the client away from these whenever possible

We should start with a set of best practices in mind, then, for each feature in the system, explain them to the client, and get their feedback. Clients will usually not be shy about saying what could work for them and what needs to be changed. We'll talk about those changes later on in this chapter. For now, we'll focus on requirements, staying close to the best practices.

Once we understand how the client's needs align with the system's defaults and main features, the consulting team can start putting the configuration together in the client's NetSuite account. The team will work with selected members of the client's **subject matter experts (SMEs)** to do this, always getting more feedback and involving them in the process as much as possible.

When all the settings, fields, searches, forms (screens in NetSuite), and so on are configured to match the client's needs, we can run them through a test with a small, select group of users. This is sometimes called a **conference room pilot** or a **walk-through**.

The aim of a conference room pilot is not to explore all possible ways to use each feature of the system; rather, the goal should be to prove that we can make it work if we stick to the *happy path* – the simplest, most common approach to each feature. As we go through that exercise with this small group, we are certainly paying attention to their feedback, watching and listening for any suggestion of friction in the process or steps we need to add to or remove from the procedure to make it successful or easier for the users. In some larger implementations, we might need to hold more than one conference room pilot, but generally, one set of sessions is enough to gather the feedback we need. Later, we'll make sure we have the client test all possible outcomes for each business process. (We'll discuss test plans in this section.)

Occasionally, we hit a hard-stopping issue during these sessions, but it's the implementation team's responsibility to think on their feet and propose workarounds to problems that arise or, in some cases, to recognize when a problem sounds serious but really is not that critical in the long run. (Maybe the use case mentioned only happens once or twice a year or is already handled as a manual process exception.)

Around this same time, we will work with the client's team to start writing out test cases and plans. A **test case** is a specific set of steps someone can complete in just a few minutes, which tests something users will need to be able to do as they perform their jobs in NetSuite. These are usually assigned to individual users who already know how to perform these tasks in the legacy system.

A **test plan** is a set of test cases that cover a function, such as *Sales Order Creation* or *Receiving Items in the Warehouse*. These are usually assigned by the department to groups that own the same processes and procedures.

Assuming we get through the walk throughs well, we can then schedule full user acceptance testing (UAT). This is where we get groups of actual users from each department within the organization, set them down with the test plans from earlier, and let them run through each test case as far as possible. These tests should include what I referred to as the *happy path* earlier (the expected process with the normal outcome) and also the *unhappy paths* (in other words, what should happen when an unexpected outcome occurs). In all cases, the testers should take notes on their experience and someone should gather all critical issues noticed or ideas for improvement for later review.

We might schedule UAT to last just a week for a small organization, or it might take a month when there are customizations and integrations involved, or when there are a lot of users who need to test the various functions.

As all of these steps are occurring, you'll need a way to track problems and suggestions for changes somehow. We usually do this via a list of *Action Items* or *Issues*. This kind of thing can be set up in a file – a spreadsheet or a document – but it usually works out better for all participants if we use a custom record in the NetSuite account for this purpose. Making a custom record with just the right set of fields, such as a *Reported by* field for the person who created the issue, *Assignee* for who it is assigned right now, the priority, and related business processes, is fairly easy in NetSuite, and having the list there allows everyone to interact with them. Take the time to set one of these lists up as soon as you think it makes sense. I usually do this once the client's people start to get hands-on practice in NetSuite so they have a place to record any problems they run into or suggestions for improvements. The project's PMs can review that list on a regular basis to make sure all issues get assigned to someone who can resolve them, and they are closed in a reasonable amount of time. If your account doesn't already have a list called *Action Items* or *Issues*, you can create these for the client and show them how to use them.

Applying your methodology to the project plan

As we discussed in the previous chapter, no project plan ever really survives unchanged to the end of the work, so knowing how to adapt to changes, how to prioritize every new request from the client, and so on, is very important. While this is the most important thing for a PM to know how to do well, every member of the implementation team needs to help with this process every day during the project.

Changes come from all sides and for all imaginable reasons. In some projects, it will be the client who tries to change most of the project's trajectory, for reasons such as changing business needs or new priorities, ideas that occurred to them only recently, or just poor planning on their part. We (the consultants) should always remember that companies moving onto the NetSuite platform are not usually experienced in making this kind of transition, so we can't get upset when they change their minds from time to time. This is just one of many things we get used to when providing consulting services.

As you work with your clients, keep what I said in mind: even if they are a software company, the people you're working with are probably not experienced with rolling out, configuring, testing, or deploying software on a large scale. Remember this and treat them with patience, since they may not always know what you mean when you start talking about scope, deliverables, test cases, or UAT.

The consulting team can be the source of changes as well. I've worked on implementation projects that had multiple PMs and consultants assigned over the life of the work, for instance. Sometimes, when new people come on board, they might have what they think of as better ways to solve issues the client has raised. Or, sometimes, someone suddenly needs to take time away from work, and that can throw off the carefully planned delivery schedule.

Whatever the source of the change, you and your co-workers must always know how to adapt to it in a way that doesn't jeopardize the overall success of the project. Here are a few ideas to keep in mind, then, when curveballs are thrown your way:

- First and foremost, the consulting team should always look for a way to solve every issue using the simplest possible approach. That usually means we start by looking at the native NetSuite feature set for a way out of any problem.

For instance, if we can solve a client requirement by configuring a new custom field onto a native record, such as fulfillment, then that's easy and quick to do.

- There are so many ways to configure custom options in NetSuite screens that it is almost always possible to find a solution, even if the team's creative problem-solving is tested to the limit. If a native feature isn't exactly what's called for, then we should talk to the client about other ways to approach the problem – workarounds.

For example, if they are used to handling certain customer requests in a particular way but you know NetSuite doesn't support that out of the box, then you should ask if they could change their process to work in a way that NetSuite can handle.

I've had clients who said that customer invoices must be printed at the time each order is prepared for shipping and that the print-out must be included in the box with the product, or at least on the same truck. We dug into that as a requirement, and they confirmed that this is really only done because "that's how we've always done it." I helped them realize that as they transition into NetSuite, they could let their customers know they would be offering much-improved electronic invoices and that this would remove the need to print invoices entirely.

- If the first two approaches here don't seem to fly, we should look for a NetSuite-provided or partner-developed custom solution that might fit the bill instead.

The SuiteApp marketplace has lots of bundles available, some free, some available for a one-time charge, and some requiring a subscription. It will make sense and save you time if you can familiarize yourself with these options on a regular basis.

The last resort should always be considering the creation of new customization within the account, but this is usually required when the client's business process strays far from the best practice and from what native NetSuite can offer. We push this option to the end of the list because building customization can be time-consuming and has a cost associated with it, and it requires the client to incur some technical debt.

One reason I say this is that every customization changes how NetSuite works from what's written up in *NetSuite Help* and *SuiteAnswers*, so users won't be able to refer to those sources any longer to understand how a given feature will work. Also, every script or workflow created to solve a custom requirement adds complexity to the system overall, which must then be monitored and maintained by the client's administrators the entire time they are using NetSuite.

Client requests received during the configuration, walk-through, or UAT phases of the project can come in many forms. Some will be simple, so we always look for the smallest solutions. Some will be more complex, though, and they can require additional time just to make sure we completely understand what is needed and research the possible ways to solve them.

Next up, then, we will learn how to deal with changes, since we know they can always occur in many different forms. Some changes, such as when a member of your own team can't finish their work on the project for some reason, you will just deal with, but many changes require coordination with the client.

Living with the plan as changes occur

Let's explore how change requests can be received from the client and how we should handle them. Knowing how to recognize one as it happens and how to respond is very important for the consultants talking to the client. We might be on a call with some of the client's SMEs, talking over how a feature will work in NetSuite, and someone from the client team might chime in, saying "Yes, and we also need to be sure that we can also track (something you've never heard of)." A good consultant is a really good listener, so mentioning a new tool they use or a new process they follow should always sound like a little warning bell in your head. I've noticed this can happen at any stage in the project, even very late in the overall schedule, so there's no time when a consultant can let a phrase like this go unnoticed.

Instead, when a client drops something new into any conversation, it's always important to explore it, understand the highlights of that topic, and decide quickly whether further investigation and action are needed. For example, say you're on a *requirements call* with the client, you know they market a line of high-tech analysis equipment to engineers, and you're asking how they handle product returns.

They talk for a while about how they process returns now, all of which you know can easily be handled with native features, but then they mention that they also sometimes send repair people out into the field, if the customer is nearby, to fix or replace equipment at their customer's site. Everything you hear from the client should be noted at that moment, but also, a few follow-up questions are in order.

Every consultant must be a good listener but knowing the right questions to ask at moments like that is what makes a *great* consultant. The **Five Ws** should apply here, so we need to ask questions such as the following:

- *Why* are they sending people out into the field, as opposed to letting the customer send the product to them?
- *Who* are these repair people within their company?
- *Where* might they need to travel to?
- *What* kind of tracking of this work do they do now?
- *When* does this kind of thing occur?

Their answers will help you to discern how big of a deal to make of this, but you will always have to offer an approach to every client request.

Using the notes at the start of this section as our guide, we'll look for ways to use native features to meet requirements like this. For this product field repair example, once we understand how it all works now, we might talk about how we can use case records to track the service calls, define service-type items to include on whichever transaction is appropriate (a new sales order or a cash sale, for instance), and then ask about whether the service is delivered for free, under a warranty, or is charged for.

If some aspect of what the client described sounds like it would be difficult to fit into native NetSuite features, we can note that at first, but generally, I avoid talking about solutions too much in this case. Instead, we should first make sure that all key stakeholders from the client agree this is a problem that must be solved, and they should agree on its importance as well. We only want to spend time exploring non-native NetSuite options, such as partner bundles or customizations, if the need is really high and there are no workarounds available.

This is exactly the kind of thing that can come up completely out of the blue, even later in a project, but when this happens, don't let it throw you off. Always keep the project's goals in mind as you work through issues like this on a day-to-day basis. Learning to deal with conversations like this in a direct way is a sign that you're an experienced NetSuite consultant.

Just follow the standard rules – remind the client of what their stated goals are, make sure someone is keeping notes on everything said during calls or on-site visits, and follow up on anything that didn't get totally resolved during each conversation.

Summary

In this chapter, we learned how the NetSuite implementation project plan drives the project teams to success, what goes into a plan, and how we apply our methodology's thinking to the plan throughout its execution.

If you've followed along with this chapter, you should now understand more about starting a NetSuite implementation project and all of the typical steps in that process. You should also have a better idea of how PMs keep a project on track and manage changes as they occur throughout the duration of the work. There are many excellent resources online and in other books covering these topics in more detail if you feel you want to dig deeper and extend the learning from this chapter.

In the next chapter, we'll dig further into exactly how we go about gathering requirements for specific parts of the NetSuite product, and I think you'll start to understand how much there is to know and how many ways a project can go awry if you're not careful. Do this a few times and you'll get the hang of it in no time.

Self-assessment

Here are some things you can think about from this chapter's topics:

1. What's the most streamlined project you've ever worked on? What worked well and what helped your team complete the work so quickly? (Try to keep those things in mind as you work on your NetSuite implementation projects since, very often, things that work well for one project will help you with any other project.)
2. Would it be a good idea to tell your client that the project goals you establish at the beginning of the project are set in stone and are the only goals you'll have for the entire set of work?
3. How will you remind your clients of the goals at the end of the project and show them how your team measured up against them?
4. Have you ever started a set of work by asking "How do you want this to work?" How did that go? How did the project go?
5. Imagine you're working with a client who is constantly throwing out new requirements throughout the project. How would you work to make them happy, but also keep the project on track and within its budget?

Part 2:

Understanding the Client's Organization

To begin a new project, we spend time talking to the client, understanding their requirements, who their users are, and how they want to organize their accounts, customers, and transactions within their new NetSuite account, all of which will be covered in this part.

This part has the following chapters:

- *Chapter 4, Documenting the Organization's Requirements*
- *Chapter 5, Analyzing the Organization's Users and Roles*
- *Chapter 6, Understanding the Organization's Accounting and Finance*
- *Chapter 7, Getting to Know the Organization's Customers, Other Entities, and Items*
- *Chapter 8, Identifying the Organization's Main Transactions*

4

Documenting the Organization's Requirements

Understanding the users' requirements is a critical first step toward completing any software product's implementation. For a NetSuite implementation, we need to understand the organization's business and people, as well as their business processes – how they're like others in the same industry and how they might differ from the leading practices.

We'll cover the following topics in detail in this chapter:

- Understanding the organization's industry
- Understanding the organization's business and people
- Use case – differences between a wholesale distributor versus software companies
- Gathering requirements and interviews
- Documenting requirements

Now that you have a good idea of what a NetSuite implementation project should consist of and you've established the project's main goals and chosen a methodology, it's time to dig further into the concept of requirements-gathering. If you've worked on other software projects, most of what's in this chapter may not be new to you, but if you are just beginning your consulting career, stay tuned. I have included lots of details and examples in this chapter to explain every concept and shed light on the process we typically follow when getting to know a new NetSuite client. Your goal with this chapter will be to learn how to gather and document your client's requirements, including all the variations and curveballs they might throw your way.

Understanding the organization's industry

Your first step in this process should be to understand the type of company you're dealing with and their industry. All for-profit and not-for-profit companies have similar traits in that they all do some sort of internal accounting, they hire people to perform their day-to-day tasks, and they must report on their business activities to one or more levels of government. Knowing the industry they fit into can help you quickly learn a few additional things, including whether they provide products, services, or both to their target market, whether they are selling things **business-to-business (B2B)** or **business-to-consumer (B2C)**, and so on. With that knowledge, you can then quickly assess which NetSuite features they are going to need and which features you can skip over.

There are many lists of well-known industries online, but we typically use a shorter list when we're categorizing NetSuite clients. We use industry names (such as those shown in the following sections) to place each company into a useful category, which helps us narrow down the names.

For our purposes in this book, we'll use these industry names.

Manufacturing

Any business that makes its products itself, even if that's only part of what the company sells, might be considered a **manufacturer**. In this category, I've worked with cosmetics makers, auto parts suppliers, plastic bag manufacturers, and food and beverage producers, as just a few examples.

These companies will need NetSuite features such as assembly-type items, assembly build, and work order transactions.

If the business has more complex needs, it might consider purchasing the NetSuite-provided Advanced Manufacturing solution or looking at some of the great partner offerings available in the SuiteApp Marketplace.

Not for profit

This category is for organizations such as disaster relief groups or food aid **non-governmental organizations (NGOs)**. Companies in this category can vary widely depending on the services they provide, but they all share a few commonalities:

- They don't have customers; they have constituents or members
- They don't have sales, but they do have donations and sometimes subscriptions

These businesses need custom options to be set up in their NetSuite accounts to support these and many other concepts that make them unique. Because of these differences, I'll mention the places where not-for-profit organizations need special attention throughout the rest of this book.

Retail

Any business where the majority of its sales come from retail stores (and/or its e-commerce store) should be considered a retail company. Most retail companies today offer a hybrid of brick-and-mortar stores and online sales. Examples of this category include clothing stores, toy sellers, online-only marketplaces, and outdoor event retailers.

In NetSuite, these companies need to track their items and inventory, build a strong e-commerce store (in NetSuite or externally), and tend to have complex marketing needs for working with their customers. They might like to use the NetSuite **Point of Sale (POS)** or **SuiteCommerce InStore** solutions to manage their cash registers in stores. They're fully integrated with NetSuite's ERP features, so sales go straight into the accounting system as they occur. Many companies also choose to implement an external service for their retail front end (such as Clover or Shopify) and then integrate that data into NetSuite, which acts as the ERP back end for the stores. Some retail clients can benefit from **NetSuite Connector**, a pre-built integration option for external services, such as Amazon.

Software

For this industry, we're talking about companies whose products are all digital, such as a tax software application, an iPhone app, or a video game publisher.

They typically have no or very few physical items they sell. Instead, they focus on digital licenses and, more frequently, software subscriptions.

Most of these businesses need contract and subscription management in their NetSuite instance and digital license key tracking. Many of these clients have complex billing requirements, and some of those will consider **SuiteBilling**, NetSuite's most flexible and advanced contracts and subscription management tool.

They might want to include complex case management for support tracking and/or a knowledge base listing important things about their products for internal users to refer to.

Services

If you're reading this book, you might work for a services organization yourself. These clients have consultants, architects, or plumbers, for instance. They typically need to track a mix of projects (such as your NetSuite implementations) and sales for services and/or items.

The service might provide marketing consulting, divorce-related legal advice, or new kitchen sink installation. They might sell items, such as the sink they wish to install, or a book offering more of their legal advice.

In NetSuite, smaller, simpler businesses need project and services management features and maybe also some item and inventory tracking. Larger service operations might also benefit from NetSuite's SuiteBilling module, as the preceding software industry does. If they need a way to track many employees' expense reports and time entries, they might be interested in purchasing NetSuite's **OpenAir** companion product.

In 2023, Oracle acquired a company called Next Technik, and so now, NetSuite clients with staff who work in the field can take advantage of the new **NetSuite Field Service Management** suite of features, including things such as staff scheduling, inventory management, automated invoicing, and many other features making field services very easy to maintain in NetSuite.

Wholesale/distributor

Businesses that buy from suppliers and sell to other businesses are known as **distributors**. These are the food, appliance, or toy suppliers who buy from the manufacturers and then provide all the things the retail or other sectors might need, usually from multiple warehouses close to the businesses they sell to. Their customers are usually those other businesses, although there's nothing in NetSuite stopping them from sometimes selling their products directly to consumers as well.

They will need items and inventory tracking, purchasing, and possibly inbound shipment management, and very likely some sort of **warehouse management system (WMS)**.

Miscellaneous

This is going to be our industry name for any company that doesn't fit into any of the other categories mentioned here. Examples of this category include companies providing various rentals, investment services, and insurance companies.

This might be a good fit for a company that doesn't sell software or services, distribute products, and so on.

For example, if a client's business is selling co-insurance to other businesses, they will have specific needs from NetSuite that don't exactly match the other options' lists. Due to this, these clients will require a bit more requirements-gathering and some solutions from your team. They can be the most interesting and the most challenging businesses to implement because their requirements can be unique!

Many companies fall into what we call **micro-verticals** or smaller industries within the larger categories mentioned earlier. For instance, wineries are micro-verticals within the food and beverage space, which can be either manufacturers or wholesale distributors. A digital marketing company is part of the larger advertising and marketing space, which falls under the general business heading. The point here is that NetSuite has many pre-defined solutions ready to be used by companies in micro-verticals and those really help fine-tune the system to match their business needs.

We assign an industry to a client so that we can quickly narrow down the list of NetSuite features they will want to use. This isn't an exact science, though, and it's not critical that you get this perfectly correct at the start. NetSuite's products offer so much flexibility that you will always be able to mix and match almost any combination of features that you decide are needed for any client. For instance, you might work with a construction company that shares requirements with both service and general business companies. We try to meet about 60% or more of the client's requirements with the assignment of the industry and then know we'll handle the rest on a one-by-one basis.

You can work with the NetSuite sales team to answer any questions you have about feature pricing and availability. Just note that sometimes, older offerings are deprecated and replaced by new features, so things such as Advanced Manufacturing or SuiteBilling, as mentioned earlier, may have new names by the time you read this.

Now that we've learned about how to start categorizing your client's company, we can learn even more about what makes them unique in their markets and how the people they work with run their business.

Understanding the organization's business and people

Moving down one layer, we need to understand the specific business we're dealing with and how the people within it are organized. Most American businesses are structured, for instance, following a standard bureaucratic model, with senior management, departments consisting of managers and employees, and many roles within each department. Getting to know how your client's business is organized and where they differ from the standard model is key to implementing NetSuite correctly for them. They will have to live in NetSuite every day once they go live, so making sure it's set up to match their exact expectations is key.

In the previous chapter, we talked about the requirements-gathering process we usually follow. I'll repeat a shortened version of that list here for your reference:

- Requirements-gathering
- Data migration planning
- Walk-throughs or conference room pilots

When you first start talking to a client, you will know which industry they are most aligned with. From that, you should know the list of initial questions you'll have for them. Some of these are standard and apply to all industries; others should be industry-specific, though, based on assumptions we usually make about each type of company. For instance, you wouldn't want to ask a software company about their warehouse operations unless you heard in advance that they have warehouses.

Your goal at the beginning is to not waste the client's time, so make sure you're pulling in any information you previously gathered during the sales cycle and around the project's kickoff, including the following:

- Any **request for quotation (RFQ)** or similar documents where they spelled out their list of required features for CRM, ERP, and so on
- Any business-specific documents you can find, such as a company overview, a **responsibility assignment matrix** (sometimes referred to as a **RACI** chart), or their current public website/store
- Any other example files they've provided, such as a chart of accounts or a list of items they sell

Note

RACI stands for **responsible, accountable, consulted, and informed** – each label indicates what level of responsibility a person should have within each part of the project.

As a member of their implementation team, you need to gather and become familiar with any documents or other information you can get your hands on before the first meeting or call. These documents might include reports they consider critical to running the business or examples of printed outputs from their current system. Some companies might have also put together a **request for proposal (RFP)** when they started looking for an ERP system. This shows the client that you are taking their implementation seriously and that you can be relied on to perform your own research. They will not want to repeat their whole history and current state for you on the first consulting call, especially if they feel they've already explained that to others within your company.

Next, take the time to get to know the people involved in the project from the client. This should always start with the project's sponsor, the executive who owns the implementation for their business. This might be the CEO for a smaller company or the CIO or VP of IT for a larger group. Whoever it is, just know that this is the person who you must satisfy in all cases and who will make the final decisions about the project's direction as it progresses and about its completion when you're done. Get to know them in person (to a limited degree, if that seems appropriate, of course) and stay in touch with them throughout the project. Know your place in the project and make sure you're not bothering the sponsor with every small decision; one of their delegates should probably handle those down-in-the-weeds topics instead.

Apart from the project sponsor, the client's team might consist of some additional people and roles. Let's take a look.

Project manager

The main point of contact is sometimes called the **project manager (PM)**.

In a very small company, this might just be someone designated as the PM for the rest of the business. In other companies, this could be a contractor who has been brought in to help the business adopt NetSuite or someone who has been hired full-time and manages other projects for the company.

Whoever they are, make sure you know how to contact them and keep them in the loop for all project-related activities whenever possible. Also, if the person assigned at this level doesn't seem used to the role, we need to be extra helpful in guiding them along the path to success.

The subject matter experts

Subject matter experts (SMEs) are usually the managers of the departments that are adopting NetSuite and others to whom they will delegate configuration and other NetSuite work.

They've been selected to be the folks who will help you understand the business and its requirements and to try out NetSuite first. Your job with these folks is to instill confidence in them that you're the right person for the job and to help them learn to use NetSuite, starting now and continuing past going live. These people – the SMEs – will frequently also become the trainers for the rest of their co-workers later in the project.

They're going to see the system while it's still being configured and will help you test alternative approaches on occasion, but often, they won't be used to configuring or testing software, so you'll need to guide them through the process before they go live.

Always remember that you're trying to teach them to fish, so show them how to do things and try to avoid doing everything for them. Early on, they will need a lot of hand-holding, but by the time you get to the final project tasks, you should be able to rely on them to do 99% of everything needed in the account.

The users/testers

Once the SMEs have vetted a given part of NetSuite for use, they will usually turn it over to users/testers to get their opinions on how things work or when the time comes for the official **user acceptance testing (UAT)**.

Always keep in mind that changing to a new ERP system is a big deal for these people. The process can mean lots of additional work hours for them, and you need to do everything you can to streamline the transition.

Whether this is a small group of accountants or large sets of people from many departments, you want to make sure everything you show them is 100% ready for them before they dig in to avoid confusion, frustration, or worse.

We always want to avoid the situation where a user or a group of them gets so frustrated with NetSuite or how it is set up that they start to speak negatively about the project or their management (or yours!) to their co-workers, who may not have started to use the application yet. For some people, once they set their minds on how badly things work, it can be exceedingly difficult to change their minds and get them back on track later. They might start to derail others' activities or the entire project.

Other project helpers

Depending on the size and complexity of the implementation, the business' IT people will help with things such as single sign-on or integrations.

I've previously worked with third-party consultants who have been brought in for auditing, process improvement, or just to keep an eye on your implementation team. These can be consultants who have been working with the business for far longer than you, and sometimes, they might view your company as a competitor.

It is a clever idea to keep separate responsibilities truly clear for these folks and maintain particularly effective communication (always include the client's PM and your PM in every message sent about the work) so that you can avoid any negative interactions with them. (That might sound harsh, but it's still sound advice. Sometimes, if things turn bad, having a paper trail of communications and proof of the work you've completed is all you can provide to a client who has lost trust in your organization.)

It's also common for us to interact with representatives from one or more NetSuite partners who are helping the client go live on their platforms. These tend to be external but related solutions, such as WMSs or cloud-based payment or billing platforms.

I've worked with a couple of third-party consultants in the past, who asked whether they could sit in on our calls or take on some of the script development, which allowed them to learn how it's done. In general, if the client wants this, it should be OK to accommodate these requests, but if anything is to be configured in the account or any customizations are to be created by the other consultant, then I make sure that the client understands that the other consultant will be responsible for their deliverables, not my team.

The beginning of a project is usually an exciting time as you're meeting new people and learning a lot about what makes the company tick. Occasionally, the complexities and differences you're learning about can be a little overwhelming. Let's look at a couple of examples and break them down to make them clearer.

Use case – differences between a wholesale distributor and a software company

To help you understand how two NetSuite clients can differ from each other, let's take a close look at two fictional businesses – *Acme Industries* from Portland, Oregon, and *IntelliChime* from New York City.

Acme Industries is a wholesale distributor of hot tubs and related parts and accessories. They purchase their hot tubs from three manufacturers, but their parts come from all over the world. They have two main warehouses to distribute from in the Pacific Northwest. They've been in business since the early 1970s, have about 200 employees (all in the Pacific Northwest region), and are currently running the business on the same ERP system they started with way back then. That legacy ERP system runs on AS400 hardware and uses green-screen terminals accessed via modern PCs and terminal software. It's been customized to a certain extent by the in-house IT team, but it's at a breaking point now. They're looking to modernize all of that and take advantage of NetSuite's cloud architecture to connect their current locations, plus two more that they are planning to open shortly.

In NetSuite, they're going to need all the standard features plus the following:

- Advanced inventory tracking for multiple locations so that they can easily expand their product lines and know exactly where everything is, what's been purchased and sold, and when it's all supposed to be received and shipped out.
- A good WMS will allow them to sync up two or more warehouses and know exactly where each item is at any time. It will also allow them to control their picking, packing, and shipping better than they can now.
- Integrations with shipping partners for both **less-than-truckload (LTL)** carriers for the hot tubs and parcel carriers for the parts and accessories. Customers expect frequent updates on orders to know when to expect shipments to arrive, so this is key for any company shipping products.

With Acme's implementation, they're going to need the following:

- A lot of hand-holding as they learn to use the NetSuite UI and find answers to their questions. You will help the end users as well as the IT folks, who will usually become the NetSuite administrators after they go live.
- Help with converting the different records in their legacy ERP system into those used in NetSuite. It's common in these scenarios for new users to continue to use the older system's terms for things such as orders, receipts, and so on.

Help them get used to the NetSuite terms, but meet them halfway on their terminology, whenever possible, to help them with this transition.

- They will need data migration assistance because of the difference in records and because the two systems are so different in architecture.

We usually just need to receive data in a CSV flat-file format for data migrations, but sometimes, even generating lists like that can be a challenge for older ERP systems.

Next, let's look at IntelliChime.

IntelliChime is all new. They sell financial management and securities software to Wall Street brokerage houses, and they just opened their doors last year. They started by running their business on QuickBooks but quickly outgrew it due to their success and a well-timed article in the *Wall Street Journal*. They're looking to bring their QuickBooks data, 50 users, and rapidly increasing sales into NetSuite as soon as possible. They have no warehouses, but they do have a little physical inventory for the security key dongles they sell, along with their main application. Everything else is digital downloads and subscription-based licenses. They also have a services team that provides consulting and support to clients who need it. NetSuite can handle all of this, but only once certain additions have been made to support their specific needs.

They're going to need the basics from NetSuite, plus the following:

- Non-inventory items for tracking the software, plus some serial-numbered inventory items for the key dongles. Each of those has a unique ID number associated with it by the manufacturer, which IntelliChime needs to track as it sells them, and if they come back as returns.
- Projects will be used to track the services' work, tied into sales orders and invoices, so that the company can manage its revenue recognition.

We call these **simple projects** in NetSuite terms, with just a few resources each and minimal cost tracking needed.

- **Advanced Revenue Management (ARM)** will help with sales revenue tracking and supports more detailed requirements if the client's revenue recognition policies demand it. ARM gives them a set of records for tracking revenue on sales as it comes in and over the life of a subscription or delivery project.
- **SuiteBilling** will be added to give IntelliChime more control over those invoices as well. This will provide them with the flexibility they need to define their subscriptions, control when invoices are generated and for which charges, and offer a few additional options when it comes to sending bills electronically.

Many of IntelliChime's customers have a complicated parent/child or bill-to/ship-to relationship, so SuiteBilling will help with that as well.

The implementation team can help the business users with a few things as well:

- Since they are a new, quickly growing company, IntelliChime is still working out its internal processes. So, the implementation team can help by always reminding them of the leading practices for things such as contracts, pricing, and billing.
- They can guide them through the tricky process of setting up all this, since SuiteBilling is both very flexible and also very complicated the first time you start working with it.

We usually implement SuiteBilling separately from the ERP implementation to make sure that the best practices for the core business are in place and configured in the account before we add the billing customizations to the system. It's usually implemented by a separate team that follows the ERP implementation.

Next, let's dig into how we can gather requirements and talk to clients at this stage in the project to get the most out of everyone's time.

Gathering requirements and interviews

In any software project, understanding the users' requirements is a key step in the early process. However, for NetSuite implementations, we want to be careful about how we start to ask our questions and talk to users. You might think that the right approach is to start by asking the client how they do things now and then translate that into NetSuite terms and processes. However, doing that can lead to a lot of unnecessary customizations if you're not careful, and that means delays and cost overruns.

When gathering requirements, it's always beneficial to put yourself in your client's shoes; try to anticipate how they will receive every question you ask and everything else you tell them. If you were them and you heard the question, how might you respond? If you realize your first idea for a question is too hard to answer, you need to ask the question differently.

For instance, asking *How many expense accounts will we set up?* is bad; it assumes that the client knows as much as you do about NetSuite, but they don't (most of the time). Instead, try leading them into the discussion with explanatory comments followed by a specific question. For example, *I believe I read that in your current system, you defined about 100 accounts for expenses. Will you be carrying all those expense accounts into NetSuite?*

It's the implementation team's job to keep a clear understanding of how NetSuite works in their minds as they discuss requirements at this stage and use that knowledge to find the best approach for the client. For instance, with that understanding of NetSuite, a consultant can guide the client into using as many native features and processes as possible. This is for everyone's benefit. The client wants to go live with the minimum investment in terms of time and money. To accomplish that, the consulting team will try to get them up and running with the simplest possible processes. We only suggest customizations when the client indicates they are needed. Finding the right balance between native and custom features that allow all of this is our main goal.

We're gathering requirements at all the stages of the project, from day one until the day the client goes live on NetSuite (and sometimes beyond that date, too). How we handle requirements as we hear about them will differ from one project stage to the next.

Here are the project stage names we'll consider:

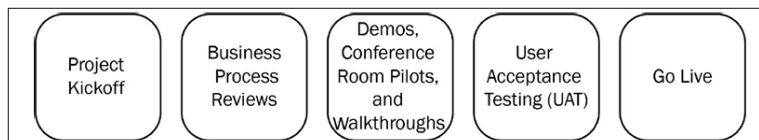


Figure 4.1 – The five stages in the requirements analysis process

We'll look at each of these stages in the following sections.

Project kickoff

Let's start with the basics and learn about the key items that will make this implementation different from any other. This will include business process reviews, demos or conference room pilots, a robust testing phase (UAT), and a clearly defined go-live or *cut-over* process.

Business process reviews

This is the main requirement-gathering time, and this is when many user interviews should happen. We'll dig into this in detail in the next section, but for now, just know that the output from this stage should be a list of NetSuite-specific requirements and your best idea on how each will be solved in the system.

How you gather requirements may vary a little from one project to the next, but generally, we use an interview format on a conference call or an in-person meeting, as appropriate. With these guided interviews, we want to help the client explain their current processes to us. They should share real-life use cases, explain which parts of their current process they need to continue with NetSuite, and add any other details they think are important. The more specific and concrete you can make these conversations, the better your understanding of their needs will be. This is where a good consultant will learn to ask good probing questions to get to the heart of what the client really needs. Listen carefully during these interviews and take a lot of notes.

We typically end up with a good list of things we can address with native features at this point in a project, and a few things we know native NetSuite cannot cover. Those will have to be solved with clever configuration or customization.

Demos, conference room pilots, and walk-throughs

After the preceding process sessions, we usually start to configure the account according to what we understand about the client's needs. Then, we need to demonstrate to the client how their business might run in NetSuite.

In these demos, we're looking for feedback on how that's working, as well as what still needs to be tweaked.

Now is a good time to introduce that **issue tracking system** I mentioned in the previous chapter. You need a place to record every issue and new request and prioritize them, and whenever it makes sense, put them on hold until after they've gone live. Also, any completely new requirements that come to light now should be called out as such, and you'll need to have a plan for how you handle them.

This is where your contract's scope comes into play – if you have a fixed bid statement of work (SOW), for instance, you may not be able to add new work to it without creating some form of **change order (CO)**. On the other hand, a time and materials SOW might be written to say that new requirements can be added once approved by the client.

Your project management and sales teams can work with the client to manage expectations around these new requirements as they arise, but in any case, make sure they do explicitly call them out.

UAT

We're fairly late in the project's overall timeline now, so new requirements should not come up often, but there's no guarantee they won't. Just as in the demonstration phase, make sure you have a process for dealing with new requirements if the client voices them.

Now is the time to start asking whether the client could live without some feature/step in the process until after they've gone live. It's usually a good idea to stress the timeline in cases like this to get the client thinking about whether they could suffer without some automation, for instance, for a few weeks, to keep the currently scheduled go-live date.

Also, most clients will change their minds about the highest priority fixes they need for their system once they go live. All of a sudden, what seemed like the most important thing before going live is much less important once users are in NetSuite every day, completing orders, and so on.

Stress this to the client at the start of UAT and remind them whenever necessary that they must keep the project on track.

Go-live

From this point forward, new requests can certainly become known, and you should have gotten the client very used to how we track them, rank them in terms of urgency, and resolve them by this time.

You should encourage the client's team to handle the smaller requests, those that might just require a new field or a very simple script. Larger requests or issues might lead to more work for your team, which may or may not be a problem, depending on your business model.

Let's dig into these business process sessions. The consultant's goal with these sessions is to quickly help the client understand how they could run their business on NetSuite in information-gathering sessions. This will partly be based on how the business is run today on the legacy CRM and ERP systems, but as I mentioned previously, you should not focus on that entirely.

So, process by process, we want to do the following:

1. Invite the subject-matter experts from the company to a call or meeting.
2. Introduce them to this process since they're probably not at all clear on what we're trying to achieve.
3. Walk them through a list of questions that will help you configure the NetSuite account for this client, but make sure your questions are tailored to what you already know about the client at this stage.
4. Help them translate their current work into NetSuite work.

We don't usually make configuration changes live during these calls, but we can certainly jump into NetSuite and show the users a screen if that helps cement their understanding of any line of questioning. For instance, when talking to the accounting and finance teams through the initial general ledger (GL) accounting setup, we need to ask about things such as their subsidiaries and chart of accounts. They might use different terms, and they might then get confused about what sorts of fields are available on both screens in the system. Showing them the screen will help give them a quick idea of what a subsidiary includes in NetSuite, but avoid walking through every field on the screen at this stage. There will be time for that level of detail soon.

As the questions are asked and answered, the consulting team needs to take special note of any new requirements that are brought up. This is very common at this stage, and the ways they can be mentioned can vary a lot as well. For instance, while talking about how sales taxes are handled today, a client might mention that they're going to need a way to handle restrictions on sales of some items in some states if the item is in some way considered hazardous or is otherwise prohibited for some people (think about cigarette sales and children, or maybe dangerous pool-cleaning supplies). It's tempting to try to spend time right then solving the issues, at least at a high level, but I generally avoid doing this. Meeting times are generally shorter than needed, and we want to focus every minute of these meetings on gathering and noting all the requirements we can instead.

Gathering requirements is difficult to learn when you're first starting. Be patient with yourself and your co-workers and clients, and you'll learn how to improve your results with practice. Capturing requirements in some sort of documentation is the next challenge we'll tackle.

Documenting requirements

How you document the requirements coming out of these sessions is up to you but write them down and make sure that all the members of your team have access to the requirement documents.

Not everyone can join every call, but if you have one consultant helping the client with *procure to pay* and another team member working on *order to cash*, they need to be able to compare notes and look for any overlapping concerns as they work things out.

You need to share these files with the client as well, of course. We commonly share files via the account's **File Cabinet** since this is a secure way to get information to all parties, and everyone has access to the account.

Next, decide on an approach that fits your overall project methodology. If you're working with the Waterfall method, you might decide to capture all the requirements in a single document (MS Word or Google Docs, for instance). Documents like these are typically referred to as **business requirements documents (BRDs)**, a set of **functional requirements documents (FRDs)**, or a **requirements traceability matrix** (usually an Excel workbook or something similar).

You will want to keep revisions of these documents as they are developed, one way or another, and then name the final version clearly. This will let everyone know that they can refer to that final version in later stages of the project.

Or, for an Agile approach, you could capture more granular user stories in a custom record in the client's NetSuite account. This approach has the advantage of allowing for great flexibility in terms of prioritizing how solutions are built out to each user story and allows for easier changes over time (versus a single, monolithic document).

Just remember everything we said in the previous chapter about the challenges this kind of flexibility can bring and only go this route if you are sure the client is 100% on board with this approach.

I've used both approaches in my project work, and I can say that having the requirements well documented before you move on to building solutions for each is critical to the success of any project. You could not imagine building a skyscraper without a blueprint, and the same applies to complex software projects like these.

The following screenshot of NetSuite shows an example of a **Requirements** tracking record from the free, NetSuite-provided **One Implementation Bundle**:

The screenshot shows the NetSuite interface for a Requirements record. At the top, there's a navigation bar with tabs like Activities, Transactions, Lists, Reports, Analytics, Documents, Setup, Customization, Implementation, SuiteApps, and Support. Below this, the 'Requirements' form is displayed. It has a 'Save' button and 'Cancel' and 'Reset' buttons. The form is divided into two main sections: 'General Information' and 'Requirement Description'. The 'General Information' section includes fields for 'CUSTOM FORM' (set to 'Requirement (Shared)'), 'ENGAGEMENT', 'IS A ONEWORLD ACCOUNT?' (checkbox), 'REQUIREMENT NAME', 'REQUIREMENT TYPE', 'PROCESS', and 'MODULE'. The 'Requirement Description' section has a text area for the requirement description and a section for the 'NETSUITE SOLUTION'. On the right side, there's a sidebar with fields for 'STATUS', 'IS A GAP' (checkbox), 'GAP TYPE', 'REQUIRES TEST CASES' (checkbox), 'PERCENT OF COMPLETION' (set to 0.0%), and 'VERTICAL'.

Figure 4.2 – An example of a NetSuite Requirements record

This bundle contains many useful record types for implementation teams to use and is freely distributed by Oracle NetSuite.

We've even used a hybrid approach in a few projects, where we started with long, complete requirements documents but then transitioned to user stories for small things that arose later in the project. However, this should be reserved for larger, longer-term projects because this can lead to some confusion among both the client's team and yours if it's not carefully managed over time.

Summary

In this chapter, I explained how to understand a client's industry and their specific business at the beginning of a project and how we gather and document their requirements at all times. Handling these requirements, even when they change the project's direction and focus, is a big part of your job. Learning to ask the right questions and listening to the answers to gather all sorts of clues regarding the client's real intent is one of your main skills as a consultant. As you improve these skills over time, you will become a more helpful member of the implementation team. The more practice you get documenting and communicating what you heard from the clients' users, the better your results will be in the long run. Your projects will be much more likely to end successfully once you've mastered these things.

In the next chapter, we'll dig even deeper into understanding the people who run the business, including management and everyone else, and how we can identify their roles within NetSuite based on their job duties.

Self-assessment

Here are a few things I hope will help you think about how to understand your clients and how you will help them implement NetSuite:

1. Which industry would you assign to a company that sells a line of cosmetics in a few of their retail stores and in many others' stores as well: retail or wholesale/distribution?
2. Think of a time when you had trouble working with someone – a co-worker, a client, or a customer – and think about what you learned from that experience. Now, imagine if that person were the key stakeholder at your NetSuite client, in charge of evangelizing NetSuite to the entire company. What strategies would you implement to ensure that the person performs that role well? How would you deal with them if they did not?
3. When gathering requirements, which is the better way to start asking a client about their purchasing process?
4. How do you handle purchase orders now?
5. Do your employees have to submit purchase requests before a PO is created?
6. Requirements can be tricky things. What you hear from the client, for instance, isn't always 100% the same as what they were trying to convey. This is why documenting the requirements is so important and why choosing to have one consistent way you document them matters so much. How are you going to document your requirements, with the goals of consistency, ease of communication, and referencing later in mind?
7. While gathering your client's requirements, you realize that one especially important requirement has no matching NetSuite feature. Which would you prefer – to offer to customize their account to meet the need or offer an external service or application for this purpose? How can you help the client make that choice?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search this book by name.

Note: Keep your purchase invoice ready before you start.



5

Analyzing the Organization's Users and Roles

To really know how a business or any other group works, we have to understand who does the work, what they call each person's role in the various teams, and how they work together. Implementing groups of people in NetSuite by function (accounting, warehouse, and so on) is a good start, but you must also understand how they interact and who's in charge. Misunderstanding someone's role in the business could lead to wasted efforts or data security issues, for instance.

In this chapter, we will cover the following topics:

- Understanding departments, teams, managers, and users
- Grouping people with roles in NetSuite
- Creating and managing employees
- Managing users with multiple roles

With this chapter, you'll learn how to analyze the groups within your client's company and apply them to roles for use in their NetSuite account.

A quick note here – now, we're getting to the content that ties into NetSuite and its various screens. For the remainder of this book, I'll be assuming that you have access to a NetSuite account and so can look at your screens to see how they compare to the screenshots included here. Just remember that because NetSuite is very customizable and the product is updated twice a year, your screens may not look exactly like mine.

Understanding departments, teams, managers, and users

When we look at any NetSuite client company, we will usually find it organized into departments, and each of those will have a manager. This is the bureaucratic approach that came into being a long time ago, but it isn't the only valid model. NetSuite doesn't require any such structure; companies are free to organize themselves into whatever groups make the most sense to them and NetSuite can be adjusted as needed.

Here's a Forbes article describing seven possible structures: <https://www.forbes.com/advisor/business/organizational-structure/>.

Having said that, we do try to start each implementation with the best practices in mind, and we know that most companies are organized into groups. Defining groups allows us to segregate responsibilities and control access to various features based on membership in those groups. It's also important to determine the list of *non-NetSuite users* at this stage.

Here are a few examples to help you understand what we usually call these groups initially, what their purpose is, and how we plan for their needs in a NetSuite implementation.

Accounting

This includes bookkeepers, accountants, and their managers. These people will need access to accounting lists (payment methods, price levels, and so on) and tax records, and they need to be able to work with journal entries and bank records, among other things.

Accounts payable

This group usually includes **accounts payable (A/P)** clerks and their management. They are the hands-on users of purchase requests and orders, vendor bills, and credits. We usually need at least one role for the front-line users and one more for approvers of the transactions they create (this is a very common request among many groups, in fact).

Accounts receivable (A/R)

These order-to-cash folks handle quotes, cash sales, and sales orders. They also need access to create bills (invoices) and statements for their customers. Like any group, we can define them as working with just one subsidiary or department, or multiples as we need.

Administrators/upper management

This group includes NetSuite administrators, the CEO, the CIO, and so on. Administrators need to be able to tweak and reconfigure most of the NetSuite account when necessary. The CEO and other upper management roles are usually granted read-only access to most transactions and other data, so they can review the business but cannot usually edit others' work.

Information technology

The IT people in most organizations need to be able to monitor others' activity, configure things such as customizations and integrations, and manage imports for things such as **items**. They are usually not allowed to edit transactions, but they might need permission to provide this kind of oversight in some cases. This group usually includes developers, and sometimes testers as well.

Sales

This is the group for inside and outside sales reps, their managers, and upper managers too. We usually see these folks divided up by region or other market definitions (such as product lines or customer groups). Their managers need to be able to review, approve, and report on their transactions.

Support

The people who provide support within the company will vary depending on the business. These are usually customer support agents of some sort (who might also sell the **knowledge base**). They sometimes also need to edit orders and other transactions.

Warehouse

Assuming the client has folks who work in a warehouse of some sort, they will usually need their own roles to allow them to receive items as they arrive and fulfill them as they are shipped out. You might want to configure roles for warehouse managers as well, in cases where you want to restrict some users from having the ability to edit these transactions or to allow managers to provide approvals before warehouse operations can be completed.

As always, your job will be to amend this list to make sense for each of your clients, and to determine how best to adjust everything in NetSuite to suit their business needs. We will start to do that next with roles, which allow us to define grouped permissions, dashboards, and more.

Grouping people with roles in NetSuite

Our goal in understanding a client's people and organization is to allow us to divide them up into roles in their NetSuite account and then assign permissions and other settings to those roles. Since security is a critical element of running your business in a cloud application, we want to make sure that each user with a login to any NetSuite interface has only the permissions they need.

We can model pretty much any business structure we need in NetSuite, via the feature known as **roles**. Roles in the application have permissions and establish a set of preferences and default views, and we use them to avoid configuring the account specifically for any individual. In other words, even if you know there is just one person who will have a set of permissions and preferences, it's best to set up a role for them.

The groups shown in the previous section are typical, but again, we're not forced to use them in NetSuite. We always start configuring roles in NetSuite using a set of predefined entries. We make copies of those when we have small changes to make to the default role, or we can instead start a new role from scratch and completely define the set of NetSuite features it will have access to as we need.

Since it's much faster to start with the NetSuite-provided roles, it will help you to know what they are and how they're meant to be used. Take the time now to review this in a NetSuite account you have access to, and you'll see they closely mirror the groups mentioned earlier. So, depending on the company's needs, these default roles can be customized, and that's what we usually do. Refer to the following screenshot:

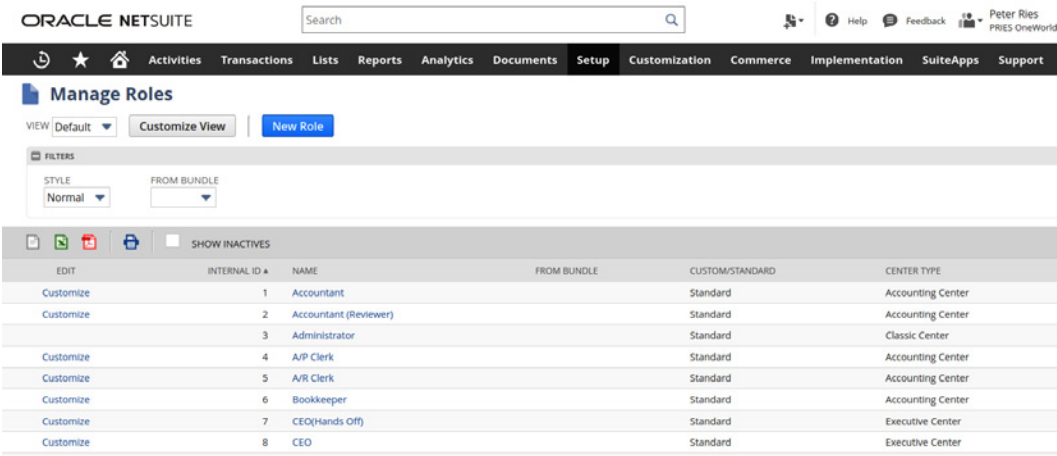


Figure 5.1 – The Manage Roles screen in NetSuite

We click on the **Customize View** link on the left (as shown in the preceding screenshot) to create a copy of the default role, and then we can change that any way we need. We name that new role after the company's name, so if they are called ABC Inc., we would name the first role ABC – Accountant. We then repeat that process for every role the company needs to assign to one or more people and get used to managing their permissions and other settings via the role, versus the individual employee's permissions.

When we start out working with a client, we don't spend a lot of time setting up and fine-tuning roles, permissions, and so on. Our main intent with this activity should be to get the client familiar with how roles work in the product and to assign roles to the first few people who will help get things started. This will allow them to start learning first-hand and to start configuring their own account. That usually means this first group of users will have more permissions than they'll need after going live, but that's OK for these very first people. Just make sure all users are clear on how this will work – also, be sure to switch users over to their proper roles before **UAT** starts. You definitely need to test the permissions and settings assigned to each role at that time, to avoid confusion or problems that only come to light when the client goes live.

NetSuite includes a default role known as **Administrator**. This role has access to everything in the account – everything! For this reason, assigning this to all the initial users is a convenient way to get everyone active in the account as you first start out. But don't do it! This can easily lead to people making changes they should not make, and the users will also get too used to being able to do anything in the account they wish to. As I said in the previous section, there should only be a very small group of people with the Administrator role in any NetSuite account, and that's true from day one.

Instead, get each user into the role they will have to learn to live with as early as possible. You will quickly learn when and how their roles need to be adjusted as the project unfolds and they test various features. This approach also shows users that security is important and gives the people who will be the administrators of the account a chance to learn how to do their jobs. This is a good example of a time when you can teach them to fish rather than doing the configuration work for them.

When it comes to deciding which permissions each role will need, quite a few factors come into play. I have explained them in the sections that follow.

Does the user need to view, create, edit, or fully manage a given record or list?

This will control which permission level you assign to the role. For instance, an A/R clerk might just need to create and edit sales orders, but the A/R manager should also be able to delete a transaction in rare cases.

Which custom records does the role need access to?

When you add custom solutions to the account, either via **SuiteApps** or other customizations, they frequently include custom record types. These are essentially custom tables of data stored in the account, and it's very common to forget to apply permissions to these tables for each role. For instance, if you create a record type called `Case Information` and want only some users to be able to access these records via sales orders, you will need to assign permissions for only certain roles to be able to view the records on that screen. One way we control permissions is via the **Permissions** tab on the custom record type's definition screen. We can also change the permissions for any single custom record type on the definition screen.

Which reports will the role need access to?

Going along with the transaction, list, and custom record permissions, we control access to reports so that users cannot gather information they wouldn't otherwise be able to read. This allows for some flexibility as well since we might say that a sales rep needs to see the report showing the total number of new cases this week, month, and year, even if they cannot see the details in the cases beyond their own department.

Roles in NetSuite also have functions beyond permissions. They allow us the following additional configuration options:

- Applying restrictions by segments such as department, class, and location, as well as custom segments
- Setting up preferred forms so that users in each role see just the fields on each NetSuite screen that apply to their work
- Setting up preferred searches for the same purpose
- Applying preferences, such as how PDFs will be handled or whether **inventory-level warnings** will be displayed, via the user's **Preferences** screen
- Setting up custom dashboards by role (so that every time users log in, they see just the shortcuts and portlets on their home page that help them do their jobs)

These are all important tasks to cover, but they should be straightforward enough that you can quickly mention them and then move on.

What do you want the role's dashboard to have on it?

Each role can have its special dashboard with predefined content to help them focus their efforts in NetSuite. It's not necessary to make a custom dashboard for every role, but you should look for opportunities to make any group's NetSuite activities easier by creating one, or by showing one of the NetSuite administrators how to do so. Giving people a limited number of shortcuts for only the NetSuite features they will use each day can be a helpful gesture for new users struggling to learn the system.

We can usually suggest a variety of portlets, tables, and charts that various groups in each industry might find helpful. For example, almost all users appreciate having reminders for the things they are tasked with monitoring in the system, such as items with low on-hand quantities or past-due customers. Warehouse workers want to see which orders are ready to ship out. Accountants want you to make their *Period Close* tasks easier. Everyone appreciates shortcuts to their most used features.

Which forms and searches should be the role's preferred versions?

NetSuite allows us to choose special versions of forms and searches for each role. For instance, you might have two sales order searches – one for managers who need to approve them, and another for everyone else. We don't usually worry about this early in the project, but this is again one of those NetSuite features we want to make sure the client is aware of. Then, when the need arises later, they can come in and add forms and searches to a role to tweak it for the users' benefit. We can also use the role to restrict access by form or search when that's appropriate.

As a member of the implementation team, you should always take advantage of these features to make each user's job as streamlined in the application as possible. Customize their dashboards to suit their needs and provide them with settings and shortcuts that allow them to focus on their unique job duties.

For some clients, as you start to give users access to their NetSuite system, they will ask you for help setting up **single sign-on (SSO)** as well. This usually involves connecting their NetSuite instance to an external **identity provider (IdP)** service. There are a few big, popular companies out there providing this service, but any service that can fully support the NetSuite authentication option known as **SAML 2.0** could be used for this. You should be OK just pointing the client to the relevant Help pages and NetSuite Support if they need that kind of help, but keep in mind that SSO is a special setting on the role as well. Each role is either enabled for SSO or is not. Make sure you're familiar with the highlights of this topic (by reading the Help pages) since it typically comes up very early on in new projects.

Once the roles are set up and you have users signing in with them, there are times when we need to troubleshoot the permissions assigned to each role. It's common to have someone report an issue such as the following error from NetSuite: **You need a higher permission for [XYZ] to do this**. This error always indicates a permission problem and can usually be sorted out by determining the role the user was logged in with and then reviewing the permissions assigned to that role. The errors are usually spot on and tell us what extra level of access the role needs before the user will be allowed to take whatever action they were attempting.

If a user cannot get to the page that would include this permission error, you can send them the URL to the exact page from your screen (assuming you can access it) and then they should get the helpful message on their screen.

In the early stages of the project, it's usually true that we don't know enough about every single user to be able to define which role they should receive, but we can at least start the conversation with the client and help them understand all the ramifications of assigning people to roles.

Next up, we'll dig into how we should go about giving people access to the NetSuite account via employee records.

Creating and managing employees

Each employee of a company should be set up in NetSuite so they can use the system for their own purposes. They might need to log in as ERP users, as accountants and sales reps do, for instance. Or they might just need access to the Employee Center so they can manage their time off, expenses, or some other custom record. We do this by creating **employee records** for each person needing access. You will want to work with your client's HR people to define this list and determine all the details you need to capture for each group of employees.

Just as with most early configurations in a new NetSuite account, you will want to customize the form the users will interact with first, making sure it includes just the native fields and custom fields and lists the business requires. If your client's requirements for this are on the simpler side, you can just customize the form once and then let them start creating new employees as they like. Businesses with more complex requirements, including those that will use the **SuitePeople** add-on, will need more attention here since they'll be tracking lots of details for each person and, in many cases, integrating this data with other systems (expenses, billing, sales, etc.). SuitePeople is usually only added to an account after the initial go-live for the main CRM and ERP functions.

Since an employee record in the system almost always contains information people consider sensitive – their home address, Social Security number (in the US) or another key identifier, and details from their performance reviews possibly – this is the time to determine how each field and list needs to be protected within the client's account. That involves setting permissions and such on the custom fields themselves, which allows us to control access to the data role by role. For example, a company might want to only allow senior managers to be able to view the employee's salary, or they might say only managers in the employee's direct chain can view this detail. NetSuite applies a lot of controls on other screens based on who an employee works for within an organization as well. If you've not worked with these controls before, check out the *Creating a Role* and *Setting Employee Restrictions* pages in the online Help pages.

Creating an employee record is straightforward: you can do so for one person whenever you need in the NetSuite UI or use the **CSV Import** feature to mass-create or update employees. Here is a short list of the key things you will want to help your clients address as they perform this key step in their implementation.

Grant access

This is when you will choose whether each person should have access to NetSuite and, if so, at what level (as a full user, an employee center user, and so on).

Assign roles

Each user who logs in to NetSuite can have one or more roles assigned to their employee record. Many users will just have one role they use every day, but some may have multiple (see the following section on managing users with multiple roles).

Set up their supervisor

It's usually a good idea to assign a supervisor to each employee. This is generally useful from an HR perspective, but it's also really helpful if you have one or more records in NetSuite that require approvals since those are usually based on the supervisor/manager hierarchy within the business.

Set up their expense limits and approvals

If employees submit expenses in NetSuite, then take this opportunity to define an expense amount they can submit before approvals are required and define the amount they can be the approver of, when appropriate.

Once employee records are set up in the account, there are a number of nice features that integrate them into the daily operation of the account. As emails are sent to each employee from any other NetSuite screen, those get tracked on the **Employee** tab automatically. You can also see all the transactions created by each employee and many other details (activities, events, and so on).

Managing users with multiple roles

In some cases, we might start with the idea that the people in the A/P department will be given either the *A/P Clerk* role or the *Accountant* role, but at some point, we might find a user who splits their time between both activities. We need to maintain accounting rules, such as the separation of duties, but assuming we're safe to give this person all the access they need, we can define one new role for them (including all the permissions they need) or we can assign them two of the more common roles. In the first case, they can just log in to their account each day and do everything they need. However, then you would have to maintain this additional role over time, just for that person.

With the two-roles approach, the user would have to get used to splitting up their time and logging into the account with whichever role they need to use throughout the day. This can become tedious, but it might still be better, since switching the roles then forces the user to think about what they're doing and when they need to do it. In my experience, when we decide to use this option, most users will adapt and get used to the switching quickly.

As you've seen, understanding the user groups that make up your client's business is important because it helps us model their access to the system, which allows them to get into the product and start doing their jobs. Security and limiting permissions to those who *need to know* is also important in nearly every business, so we like to be sure the SMEs and managers we train first all know how to set up and change roles in NetSuite in preparation for all the configuration and data creation to come.

Summary

In this chapter, we've covered how to begin the conversation with a client about configuring roles in a NetSuite account, and some of the things that can be tricky when implementing them. Taking the time to find out about any groups of employees the client has at this stage (even if they won't be NetSuite users) will prove valuable for you in the long run since they can still affect how you configure the account in terms of security, reporting, and more. You also learned how to create employee records for each user and how they should differentiate people by role, department, and so on.

Don't forget what we said about handling users with multiple roles or those who need a new custom role to be created just for them. We usually start this process with each client early on in their implementation and then come back to it and tweak the plan as needed, right up until going live. Leading your client through this process gives them confidence in your abilities too.

Now that we have this overview in mind and a few people are assigned to a few roles, we can dig into more meaty configuration tasks. So, in the next chapter, we will start to configure an account for finances and accounting.

Self-assessment

Here are a few things related to roles and permissions for you to think about when you're implementing a client:

1. If you're working with a client who is very new and doesn't have departments or groups yet but tells you that they want to start setting things up in NetSuite according to best practices, how can you guide them to the right solution, without creating too much extra hierarchy and so extra work for them now?
2. While implementing a client, John, a senior product engineer, tells you that he needs to have the *Administrator* role to be sure he can get his work done and *adjust anything he doesn't like about how NetSuite is set up*. How should you respond, and who else from the client's implementation team should be involved in deciding how best to help John work in NetSuite?
3. As you are helping a client import their `Employee` list for the first time, the client asks whether they can attach a custom record to employees, tracking each person's results in the annual football pool. Nearly everyone in the business participates, and the competition is a highlight of the year. What's the simplest thing you can do to help your client manage this, without spending too much time on it?
4. If we say that we won't get every role set up perfectly at the start of a project, when exactly do you think they should be set up? It could be that you could do this just before the UAT starts, or you could put it off until just before going live. There are certainly advantages to waiting, but what might some of the disadvantages be?

6

Understanding the Organization's Accounting and Finance

Every business, including not-for-profits, has some accounting to be done in NetSuite. However, since not every NetSuite client runs their business directly in the ERP system, determining just the right mix of accounting features your client's organization should use is an important skill you will need to develop. This starts with understanding NetSuite's features and the terms used to describe them but also requires an understanding of the laws and regulations for the region the business will operate in, along with other factors. By the end of this chapter, I hope you'll know how to do all these things and get your client on the right track to maintaining their accounting and finances in NetSuite.

In this chapter, we will learn about implementing the following NetSuite features:

- Enabling features and setting up the accounting basics
- Defining the general ledger and Chart of Accounts
- Setting up segmentation – subsidiaries, locations, departments, and classes
- Understanding the Accounts Payable feature
- Understanding the Accounts Receivable feature
- Use budgeting to plan and control finances

From this chapter forward, we will mention at the beginning of each chapter who you will be working with as you cover these topics during the implementation – in other words, who the subject-matter experts are.

In this chapter, you will be helping the senior leaders and the accounting department of the business to get started with the product. Make sure the users who will be making these configurations have the right permissions in the account (via their roles) to streamline these sessions.

Enabling features and setting up the accounting basics

When we begin to work with a new client, their NetSuite account is created from a template by the NetSuite infrastructure team. As it comes online, it has its own unique account number and URL, so we can then log in to it. This initial account is what we call the *out-of-the-box* NetSuite product. It has some default features and settings enabled, but generally, it's not very useful yet. It requires us to begin tweaking those defaults and enabling more features to suit the client's business needs.

If you're reading the chapters in this book in order, by the time the account is enabled and you have access to it, you should have already begun generally understanding the business and gathering the client's requirements. Now, you can start to walk them through the initial setup of their account, and that usually starts with setting up the *company information* and *enabling features*.

On the **Company Information** screen, you can enter the business's **COMPANY NAME**, **LEGAL NAME**, **COMPANY LOGO**, **RETURN EMAIL ADDRESS**, physical address, and more. Refer to the following screenshot for an example of what this screen looks like:

ORACLE NETSUITE

Search

Help Feedback Peter Ries PRIES OneWorld Account - Administrator

Activities Transactions Lists Reports Analytics Documents Setup Customization

Company Information

Save Cancel

COMPANY NAME *
PRIES OneWorld Account

LEGAL NAME
PRIES OneWorld Account

COMPANY LOGO (FORMS)
pries.png

COMPANY LOGO (PAGES)
ts-dev.jpg

☐ DISPLAY LOGO INTERNALLY

WEB SITE
https://peterries.net

COUNTY/STATE/PROVINCE *
Delaware

COUNTRY
United States

RETURN EMAIL ADDRESS *
pries-return@netsuite.com

FAX

CURRENCY
USA

EMPLOYER IDENTIFICATION NUMBER (EIN)

SSN OR TIN (SOCIAL SECURITY NUMBER, TAX ID NUMBER)

☐ PURGE ACCOUNT

FIRST FISCAL MONTH *
January

TIME ZONE
(GMT-07:00) Mounta...lme (US & Canada)

ACCOUNT ID
TSTD RV2081801

Figure 6.1 – The NetSuite Company Information screen

This is a good first screen to walk a client through in that it has relatively few fields and a couple of sub-lists, so they can get used to browsing the NetSuite UI. This is a good time to talk to users about how they can usually change a field's value once it's set, but not always. Since the NetSuite UI doesn't call that out, it will be up to you to mention the fields, lists, and others where this applies. You should also know that for OneWorld clients, many choices you make that apply to the parent subsidiary will be applied to the child subsidiaries as well.

Next up, introduce the client to the **Enable Features** screen. This is where we like to talk about how some features can simply be turned on here, but other features require NetSuite to enable them first. Those are the billable modules and add-ons that have a cost associated with them. The client's sales team should have already enabled the modules that were agreed upon in the sales cycle, but it's always a good idea to review these things as you complete these first NetSuite walkthroughs and confirm everything that should be available in the account is enabled correctly.

Some of the features you can enable here have specific legal terms and conditions that apply to them, and those terms must be accepted before the feature becomes enabled. Only a representative of the client's company with the relevant authority can accept those terms. For this reason, someone from the client company should be in control as you explain this screen, so they can click the **I Agree** button each time it pops up.

On **Enable Features**, we don't usually talk to the client about every single option – there are a lot! Instead, have a plan in mind to talk only about those options you know are relevant to the business. For instance, if they are only using the financial features of their NetSuite account, then you can disable **accounts receivable (A/R)** features such as Sales Orders and Return Authorizations. There are many basic features and several advanced options too. You can usually equate *advanced* with *more complicated*, but this also means *more powerful* too. I generally only enable these features if I am sure a business needs them, as making their account harder to use is never seen as a positive. This is where experience with the system over time pays off for you as a consultant or NetSuite administrator.

In any case, go through enough of the features to make sure the users are familiar with this screen and know to come back to it if they need to tweak their account. This is the time to highlight any features that cannot be disabled once they're enabled (such as Advanced Revenue Management) and to explain how, sometimes, when you enable a specific feature, NetSuite will let you know that it needs to enable another at the same time. This is also a good time to reinforce the idea of roles and permissions, and to point out that only NetSuite administrators and custom roles with this specific permission can access this screen.

Lastly, this is also the time to introduce the idea of *change management* to the users. A NetSuite account is made up of thousands of settings, all of which have a material effect on users' experiences with the system. As you've seen, we start tweaking those things from the first day, and the pace of change only accelerates as we approach the *go live*. Keeping track of changes is not critical when we start out, but the closer we get to going live, the more important it becomes to keep track of the big changes we make in an account. Make sure the users are aware of this first and then make sure everyone (not just the first users) is aware of the importance of only making changes when the client's group of administrators

has OK'd them. Make sure everyone knows how to review the System Notes for each record and the other historical data NetSuite makes available too.

If you're not familiar with them, **System Notes** are a built-in feature in every NetSuite account on every screen that represents a record of some sort. The system automatically adds a System Note every time the record is created or changed. It also tracks the deletion of records, but that history can be found in a separate search, known as **Deleted Records**. System Notes come in two styles now, as the system is slowly transitioning from an old, somewhat less informative style to a new *version 2*, which includes additional details. Regardless of which style a record has, you can always find System Notes on a subtab named something like *System Information*, but access to these details is very dependent on the permissions your role has for each record type.

Once the most important features are enabled and the client's first users have seen how you do this, you can move on with confidence that the screens you need to set up next will all be there and will work as expected. After the meeting where you walk through these steps with the client, it's usually a good idea to review all of them to make sure they were done correctly and to close any loose ends or open questions that came up during the session with the users.

Next, let's talk about the next set of setup steps we usually take with new clients – the general ledger and accounting information.

Defining the general ledger and Chart of Accounts

Every client I have worked with within NetSuite wants to use its financial features. I suppose it would be possible to only use NetSuite's CRM features and not the general ledger (GL), but I haven't seen this done yet. That means they're going to want to set up their COA next (I'm assuming here that you have a basic understanding of business accounting as we cover this topic, by the way).

The Accounts list in NetSuite is found at **Setup | Accounting | Chart of Accounts**. Out of the box, NetSuite includes a list of the most common accounts most businesses will use, but you'll find you don't use that list very often. More commonly, the business will have a list of accounts in its current system and, for the most part, will want to use the same account names and numbers. This is OK, as you can help them use the **CSV Import** feature to bring their accounts into the system easily (in fact, that's usually the first thing we teach them to import, and it's a great skill for them to have since we'll have lots of other lists and data to import very soon).

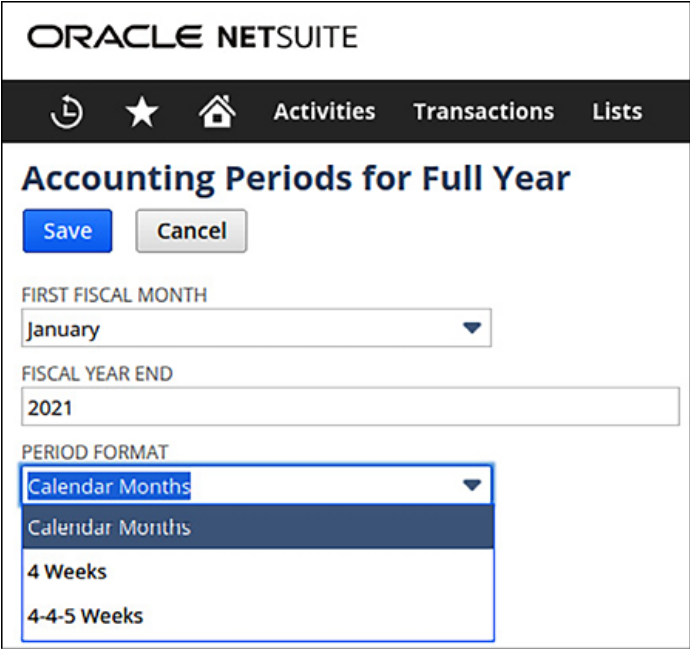
As you work to define this list, the users will notice that NetSuite's **Account Types** list is static, meaning you can't add a custom value to this list. That should normally be OK, as we use this occasion to remind the client how we're implementing with best practices in mind, and it really shouldn't be necessary to add a new value to this list.

As you talk about each of the available types, you can expand on things as needed, so briefly talk about bank integrations for bank accounts, and expense categories for expense accounts. Talk about currencies while you're there, and how NetSuite can pull in exchange rates from the internet if they need to.

This is where we start getting to some chicken-and-egg scenarios as well. For instance, to fully define the accounts, you might need to apply restrictions by class, department, or location (the segmentations), but you might not have covered them yet. Just mention these things when you get to them and make a note that you'll come back to them to finish the definition of the account once you have those other things in place.

Assign someone the homework of getting their COA into a format that can be imported into the system. It's a good idea to provide a client with a template, maybe in a text CSV file or as an Excel file, so they don't have to start from scratch. Help them go through that since it's probably their first time using the CSV Import feature, and it can be a little tricky at the beginning.

Also, before we move on, this is a good time to introduce users to the **Accounting Periods** screen. This is where you'll have a conversation with them about how they will track their financial periods for reports. Some companies make this easy and use a standard calendar year, but NetSuite offers a variety of choices here for companies, depending on how they wish to set up their periods. Since most companies will have made this choice before and want to continue using the same model they are using now, you'll help them choose from the available options. You can see them when you select **Set Up Full Year** and, on that screen, you select the **FIRST FISCAL MONTH**, the **FISCAL YEAR END**, and the **PERIOD FORMAT**, as shown in the following screenshot:



ORACLE NETSUITE

Activities Transactions Lists

Accounting Periods for Full Year

Save Cancel

FIRST FISCAL MONTH
January

FISCAL YEAR END
2021

PERIOD FORMAT
Calendar Months
Calendar Months
4 Weeks
4-5 Weeks

Figure 6.2 – Setting up the accounting periods for a new year in NetSuite

The **Calendar Months** option is the simplest to maintain, but all three options NetSuite supports are selectable. If we select either **4 Weeks** or **4-4-5 Weeks**, then we want to make sure that the client knows this affects how posting periods are assigned to transactions based on their dates. This is typically exactly what the accounting team wants, but it can confuse other users later if it is not explained to them. Next, we'll dig into how to set up segmentation.

Setting up segmentation—subsidiaries, locations, departments, and classes

We use *segmentation* in NetSuite to classify and categorize purchases, sales, and items, among other things. NetSuite's Help also refers to this as *classification*. Both terms mean the same thing. This is most helpful for accounting and other types of reporting, and so the accounting teams usually have a lot of opinions for exactly how this needs to be done.

NetSuite offers the following lists as the primary means for segmenting a company's activities. We can set these records up in an account via the **Setup | Company** menu selections, and then the appropriate choice, as described here. As you do this, think about the hierarchies at play and define the base records (a parent subsidiary, for instance) before the child records (sub-locations, for instance).

Subsidiaries

A *subsidiary* is the highest level of segmentation. It represents the business's legal entities (in fact, many businesses use this term). Talk the client through setting these up on the **Setup | Company | Subsidiaries** screen and about how they'll use the system's hierarchy (with multiple levels of parent companies) and whether they'll need to use elimination subsidiaries as well (in which case, check the **Elimination** checkbox). Notice that in NetSuite, many things are defined on each subsidiary's screen, including the logos and the website's URL.

In most cases, your clients will just need to quickly learn what NetSuite means by *subsidiary*, then they'll know exactly how to tell you to set them up. Some smaller companies can just get along with one parent company subsidiary, but many companies need more. Apply settings as you go while you create their subs (subsidiaries) in the account. You can delete a sub after creating it, but not once there are any other records relating to that sub, so take the time to get this done correctly (this same rule applies to all these segmentation elements).

Locations

When we talk about *locations*, we're usually referring to offices, warehouses, or stores. But your client's business might have other locations they want to keep track of in the system. For instance, they might need to store inventory in a location that doesn't exist (for accounting reasons versus warehouse reasons). The system doesn't provide any starter location entries for us, since there's no way for NetSuite to know what each client needs here. However, getting them into the account, just by typing them in, is usually not a chore.

If the business sells software, they might not have any use for multiple locations, but otherwise, take the time to talk the client through the fields for at least one location while you're here, since we want to make sure we get this done correctly. If needed, you can define a hierarchy again for situations where a location is needed for inventory tracking (for instance) and the client wants to be able to roll up reporting for the main location and its sub-locations.

Make sure you check the **MAKE INVENTORY AVAILABLE** checkbox if the client needs to track any sort of inventory for this location:

The screenshot shows the Oracle NetSuite 'Location' setup screen. At the top, there's a navigation bar with 'ORACLE NETSUITE' and a search bar. Below that, a menu bar includes 'Activities', 'Transactions', 'Lists', 'Reports', 'Analytics', 'Documents', 'Setup', and 'Customization'. The 'Setup' tab is active, and the 'Location' sub-tab is selected. The form has a 'Save' button (highlighted in blue) and a 'Cancel' button. The form fields include: 'LOCATION IS INACTIVE' (checkbox), 'NAME' (text field with a star icon, containing 'Main Warehouse'), 'PARENT LOCATION' (dropdown menu), 'SUBSIDIARY' (dropdown menu with 'Parent Company' selected), 'LOCATION TYPE' (dropdown menu with 'Warehouse' selected), 'TIME ZONE' (dropdown menu with '(GMT-07:00) Mountain Time (US & Canada)' selected), 'LATITUDE' (text field), 'LONGITUDE' (text field), 'DOCUMENT NUMBER PREFIX' (text field), 'LOGO' (text field), 'DEFAULT ALLOCATION PRIORITY' (text field), and two checkboxes: 'MAKE INVENTORY AVAILABLE' (checked) and 'MAKE INVENTORY AVAILABLE IN WEB STORE' (checked). At the top right of the form, there are links for 'List', 'Search', 'Customize', and 'More'.

Figure 6.3 – The Location setup screen in NetSuite

This might include cases where we create a *virtual location* for tracking damaged goods within a warehouse. We do this sometimes because it's a simpler way of tracking bad products. This is a case of NetSuite offering much more advanced features if we choose to use them; however, for some companies, just being able to *store* damaged goods in a separate location is enough from a reporting standpoint (the folks in the warehouse will already have their own process for dealing with broken things). If the client will use SuiteCommerce or SuiteCommerce Advanced to sell items via their e-commerce web store, then check the **MAKE INVENTORY AVAILABLE IN WEB STORE** checkbox as well.

Departments

Most businesses will come to NetSuite with an idea of their *departments* already, though occasionally the client wants more than they need. For example, one idea is to set up one department for each cost center, and then to use them as expense segments in reports. In the screen where we set these up (**Setup | Company | Departments**), you'll notice there are just a few fields. We give them a name and define a hierarchy again, and then we assign them to one or more subsidiaries. It's a good idea to demonstrate how the system uses departments as the main way to categorize all reports. Most clients should be able to provide you with a list of departments for importing but you can always help them fine-tune that list. Once that's done, assign that task to someone so the client will perform the import.

They can choose any department names they like but do explain to them that they need to be unique across the segments they're using, and the list should be complete. So, they need to define departments to cover all their sales, purchases, accounting, inventory categories, special cases, once-per-year events, and so on. This is a *teach-them-to-fish* moment you don't want to avoid.

Classes

Every business has an idea of how they want to see their reports broken down, and some will need a metric beyond the legal entity (subsidiary), the locations, and the departments. This is where *classes* come in. Since each business can define its class list however it needs to, it can use this to indicate whatever other metric it wants to use for its reports. The most common ways we set up classes for clients is to describe their profit centers, or sometimes their product categories. For instance, a software company might offer Tax Software, Financial Planning Software, and Utilities. But sometimes, this doesn't come to light immediately, and that's OK. You are never required to create classes, and you can easily come back to them if a need arises later. Just like reports, the client should provide the list of values they want to use, and you can help them make sure that the list will work in NetSuite during the import.

Custom segments

For companies that need even more ways to classify things, the system also supports the definition of *custom segments*. We define custom segments by going to **Customization | Lists, Record & Fields | Custom Segments**. You'll need to assign the label to each segment you create carefully since they cannot be duplicates of any other segment, custom record, or field name. Segments can be named for any other metric/measure they need, assuming they're already using departments and classes. They get a custom record type for storing the data, and then they can be used on entities, transactions, and so on, just like the native segments.

Before we move on, let me point out that there are many, many other settings, preferences, lists, and more to be set up in a new NetSuite account, and we're not going to cover them all here. Make sure you talk to the client about these things, though, just because they are important to most businesses and new users will usually take a long time to find them on their own. Here are a few more screens that are used when initially setting up the system:

- **Setup | Company | General Preferences**
- **Setup | Company | Rename Records/Transactions**
- **Setup | Company | Auto-Generated Numbers**
- **Setup | Accounting | Accounting Preferences**
- **Setup | Accounting | Accounting Lists**

Another thing you might need to enable and set up now, if the business has a clear need for it, is NetSuite's **Multi-Book Accounting**. It enables businesses to maintain multiple sets of accounting books for the same transactions to meet diverse regulatory requirements, with separate journals, period closes, and consistent reporting across books. I won't go into much detail on that here, but you enable the feature under **Setup | Company | Enable Features**. But make sure you're very familiar with all the pros and cons of this feature before doing so. Seek guidance from a more experienced NetSuite consultant if you're new to this topic since setting this up incorrectly can be costly and time-consuming if you need to change the feature/settings later.

With these features set up in the account, you'll know that every bit of data you create in the account will show up on accounting and other reports just the way the client wants to see them. With this safely done, you can next move on to some of the other accounting-related features for **accounts payable (A/P)** and **A/R**.

Understanding the Accounts Payable feature

With A/P, businesses get the standard features, including the relevant transactions (Purchase Requisitions, Purchase Orders, Vendor Bills, and Returns) and settings, but you might also want to enable a couple of other features for clients who need them. For instance, **Blanket Purchase Orders** make it easy to manage bulk purchases that are received in smaller quantities over time, and the **Advanced Receiving** and **Inbound Shipment** options are available for businesses that need more granular inbound transaction tracking. One other nice feature is **Vendor Prepayments** for companies that make deposits at the time of their purchases and then apply them to the bill once the order is received.

On the NetSuite **Help** screens, you'll find most of the A/P topics under the **Supply Change Management (SCM)** heading. We're going to cover the setup of things such as vendors and items in the next chapter so, for now, there's not a lot more to be set up for A/P.

Important note

Other important topics for accountants in NetSuite are period close and account reconciliation. We'll cover these topics in detail in *Chapter 13*, but for now, just know that NetSuite just released a great new highly automated (and AI-powered) **Account Reconciliation** feature that is well worth considering for clients who are concerned about spending a lot of time and effort on this process.

Understanding the Accounts Receivable feature

On the A/R side, there is a bit more setup to be done. In addition to choosing which transactions your client will use, you can also choose whether to enable things such as promotions (NetSuite offers a few different solutions on this, depending on how complicated the promotions can be), gross profit tracking, and all the shipping options (since they mostly affect sales). Shipping can be very simple or very, very complicated, so we'll tackle it in more detail in *Chapter 15*.

If the business is in the services industry, this is when you would enable the **Projects** features, or maybe even NetSuite's **Advanced Projects**.

This is also a good time to talk about SuiteApps that might be needed, such as **Avalara's AvaTax** solution, or one of the advanced invoicing apps. As stated before, understanding the client's requirements before you get into enabling features will smooth this process considerably.

We won't go into these other things in much detail, but just know that when needed, you should be beginning to talk to the client about a few other things, such as the following:

- Taxes
- Shipping
- Manufacturing
- Sales and marketing settings

These are important topics for some businesses, depending on their use cases. Retailers are concerned with taxes, wholesalers with shipping, and so on.

Having set your client's segments and accounting features, you can be sure that NetSuite will show you the right screens and allow proper initial reports (before you start customizing, at least). This is important, and that's why we cover this at this stage.

Use budgeting to plan and control finances

Every business ought to have a plan estimating their revenue and expenses and how they will manage those things to meet their short- and long-term goals. This is what a budget is, and businesses should perform their day-to-day operations and their planning according to their budget. Without this, the business doesn't know whether they're on track from a financial standpoint.

NetSuite allows us to create a budget for the entire business or for each subsidiary (in a OneWorld account). We do that by defining a budget amount for each GL account, thereby defining the anticipated revenue and expense amounts for each fiscal year. We can create a budget in the NetSuite UI or via CSV import, or even by copying an existing budget to create a new one. The system supports budgets for multiple currencies and accounting books, and we can even define more than one budget per period and business unit if we choose to.

Once you have a budget set up inside NetSuite, you can then run reports to compare actual business activities to the budget. Your clients should do this since this is a very handy way to measure projected amounts to actuals, once the company is live on NetSuite. Most commonly, that will include the **Budget Income Statement Detail** and the **Budget vs Actual** report. Here is an example of the former report from a NetSuite account:

Budget Income Statement Detail Back to Summary					
FINANCIAL ROW	SUBSIDIARY	CLASS	DEPARTMENT	LOCATION	AMOUNT
<input checked="" type="checkbox"/> Ordinary Income/Expense					
Gross Profit					\$0.00
<input checked="" type="checkbox"/> Expense					
<input checked="" type="checkbox"/> Advertising					
	HEADQUARTERS	Direct Sales	Domestic Furniture	- No Location	\$700.00
				-	
Total - Advertising					\$700.00
<input checked="" type="checkbox"/> Insurance Expense					
	HEADQUARTERS	Direct Sales	Domestic Furniture	- No Location	\$2,000.00
				-	
Total - Insurance Expense					\$2,000.00
<input checked="" type="checkbox"/> Miscellaneous Expense					
	HEADQUARTERS	Direct Sales	Domestic Furniture	- No Location	\$200.00
				-	
Total - Miscellaneous Expense					\$200.00
Total - Expense					\$2,900.00
Net Ordinary Income					(\$2,900.00)
Net Income					(\$2,900.00)

Figure 6.4 – An example of a Budget Income Statement

New since 2023, NetSuite now provides a standalone (but integrated) offering known as **NetSuite Planning and Budgeting (NSPB)**. This is the better option for companies wanting lots of control over their budgeting and reporting on their budgets.

Special use case – fixed asset management

Some businesses have to follow accounting rules to track and properly report their accounting around what are known as *wFixed Assets*. These are things the business owns that cannot be easily converted to cash, such as buildings, vehicles, and equipment. NetSuite provides an add-on that is specifically designed to facilitate this kind of accounting, and it's a lifesaver for accountants, helping them track costs and depreciation for their assets over time, including the sale and disposition of those things. As a consultant, you'll need to know when to bring this topic up and to guide your client with this module. More info can be found on the product's website, <https://www.netsuite.com/portal/products/erp/financial-management/finance-accounting/fixed-assets-management.shtml>.

Summary

In this chapter, we covered how we usually walk the client through the initial configuration choices in their account and get things such as segmentation and preferences set up. This is key to the entire implementation since the choices we make now will have a large effect on how we can use the rest of the account later.

Most of these choices can be tweaked if things change later, but knowing about the ones you cannot change or can't delete later is important too. Apply the things you learned in the requirements and roles and user phases earlier as you help the client define their lists of subsidiaries, locations, departments, and so on. If you do this, you should have no trouble getting through this work for each of your clients. You should also have the users set up the screens whenever possible, so they get the hands-on experience they need to feel confident when working in the system.

In the next chapter, we'll learn how to analyze your client's lists of entities and items and begin to set those up in the account. For many companies, this is an *aha!* moment where they start to see how their NetSuite account will be set up for their specific business, and it's fun to be the consultant there when this happens.

Self-assessment

Here are a few things related to the initial account setup for you to think about each time you get to this point in a project:

1. Your client tells you they have multiple businesses, and they must keep them completely separate in the system. They ask you how to enable separate accounts. What do you tell them? How can you help them achieve what they need with their one NetSuite account? When would you consider the idea of having more than one NetSuite account?
2. While talking through the **Enable Features** screen, your client mentions a new requirement related to a feature that you've never heard of before, and you know this will be a significant change in the project, affecting the schedule and budgets. In addition to letting your project managers know about this, when would you bring the sales team (your company's or NetSuite's) into the conversation?
3. One day, you receive the list of departments your client uses in their current (legacy) ERP system and notice there is an entry called *Assembly*, but you've never discussed it with them before. You ask about it, and they say this is rarely used now and is just for times when they create paper documentation for their software. What's the simplest way you can set this up for them without creating a lot of extra work for the accounting people?

4. While discussing the accounting lists and preferences with the accounting team, an argument starts over some detail. The accountants want to change how expenses have always been broken down in the COA to cover their needs, but the accounting manager doesn't see the need to change this. The conversation starts to drift, and you can tell this could take a long time to resolve. How can you help smooth things along and avoid losing track of the main purposes of the call or meeting?
5. In the middle of your segmentation session, talking about how custom segments might solve a problem someone brought up, a user asks you a question you don't confidently know the answer to. Which NetSuite features might you open up and reference, in order to both answer their question and guide them to the helpful information provided within the system?
6. One day, you ask your client, a rapidly growing software company, if they want to set up budgets in NetSuite, and the CFO tells you they don't usually do this since their business is growing so fast and projections are always incorrect. How would you reply to this? Do you see any value in pursuing the budget feature for this business?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search this book by name.

Note: Keep your purchase invoice ready before you start.



7

Getting to Know the Organization's Entities and Items

With some organizations, it can be difficult to discern who their customers are and what types of items they buy and sell, but getting this right is critical to your implementation. New consultants need to learn how to get information like this from a client, and how to think about and organize these concepts correctly the first time and every time.

We'll dig into gathering the client's requirements around these topics in this chapter. We'll also learn how to configure these aspects in NetSuite in *Chapters 10* and *11*.

In this chapter, you will learn about the following:

- Differentiating client leads, prospects, and customers
- Implementing customer projects and resources
- When to use contacts versus sub-customers
- Managing vendors, partners, and other entity types
- Identifying items and their types
- When to use special-purpose items

When performing the steps in this chapter with a client, you will be helping the customer and vendor SMEs and very likely the sales team. They need your help with defining their customer and supplier relationships and tracking their competitors or partners. You'll be mapping out the details about the company's item data as well.

In this chapter, you will learn how to use **entities** in NetSuite to model the people your clients work with, and **items** to model the things they purchase, sell, and so on. When we use the term *entities*, that includes customers, contacts, vendors, competitors, and partners. These are lists of the most commonly referenced records in NetSuite, and they're just as customizable as any other record. Everyone has *customers* (though they may not call them that) and *vendors* (aka suppliers, contractors, utilities, etc.); some clients will use other entities as they need them.

Differentiating client leads, prospects, and customers

When we talk to clients about who their customers are, we usually want to help them break those lists down into the following three categories, as a starting point, so that we can easily map the lists into NetSuite:

- **Leads** are people or businesses that you haven't sold anything to yet but who you believe might be interested in your products or services
- **Prospects** are people you have started talking to but have not sold anything to yet
- **Customers** are companies or people you have a sales transaction with and whom you hope to sell to again in the future

How your client thinks about their customers might start very differently. We find this is commonly an area where terminology gets confusing, as the client uses the terms they currently use in their business, and consultants tend to use the terms NetSuite uses. As a good consultant, you must always be conscious of when you're doing this and when using the NetSuite terms might confuse users. We usually start by mimicking whatever terms the business is used to using but then help them transition to the correct NetSuite terms in time, once we know which features they will use (this rule applies to most of NetSuite's features, in fact, so keep that in mind).

For example, some companies never deal with leads or prospects, so they are all referred to as *customers*. Depending on the industry and the region they work in, some businesses will have other terms they use for their leads and prospects. Some might have another level they want to define, to further complicate things.

Learn your client's terms first, and then when it's time to start bringing that data into their account, you can recommend which entity-related features they should use and, if needed, talk about the pros and cons of using each in their day-to-day operations. For more on configuring and using those features, see *Chapter 12*.

We usually only choose to track leads when the client receives lists of people, either from marketing firms or via events such as trade shows. They typically only receive a name and a way to contact the person, such as an email address or a phone number. This is where **lead nurturing** marketing campaigns come in, and NetSuite has some nice features for clients who need them. We use that term – lead nurturing – to describe the process whereby we attempt to convert leads into prospects or paying customers, over time. We don't want to call those people or businesses *customers* yet, but we might call them *prospects* instead. This is if we don't need to differentiate between leads and prospects, and if there's some above-average level of confidence that sales will ensue. There's no reason to use both *leads* and *prospects* if the way we'll use the information is the same in every case.

However, when we talk about *customers*, we do need to be careful. Many clients' ideas of who their customers are will not necessarily fit into the way the NetSuite system stores a customer record, so sometimes adjustments in their process and the terms they're used to using are required. For instance, I once had a client who called everyone they did business with a *customer*, including their suppliers and their partners. Their legacy ERP system didn't differentiate these terms, somehow, and so they were used to using the same words that the prior system used. When situations like this arise during your information-gathering stage, take the time to educate users on how NetSuite can improve this situation by splitting up these lists.

Not all businesses have “customers.” For example, a not-for-profit company might have *donors* or *members*, and that's OK. You have two options on how to consult with the client around terms such as these. You can show them the **Rename Records** screen (under **Setup | Company | Rename Records**), but then you'll have to have a conversation about the effect this has on various screens and reports in their account. I always mention how if they do rename a record, for example, changing a customer into a member, they need to remember that the system's Help screens and SuiteAnswers won't be updated to match; so later, when they have a question about setting up members, for instance, they'll have to search for *customers* instead. This is OK, and many clients do rename records to match what their users are used to saying.

The other option is, of course, to keep the NetSuite terms in place, and then everyone using the account will need to get used to the system's terms. Sometimes this can be a tough call, but it's usually not a difficult choice to make.

The screenshot shows the Oracle NetSuite interface. At the top is the 'ORACLE NETSUITE' logo and a search bar. Below this is a navigation bar with tabs: Activities, Transactions, Lists, Reports, Analytics, Documents, Setup, and Customization. The 'Customer' record is open, with a 'Save' button and a 'Cancel' button. The 'Primary Information' tab is selected, showing fields for CUSTOM FORM (Standard Customer Form), CUSTOMER ID (To Be Generated), TYPE (COMPANY selected), COMPANY NAME, PARENT COMPANY, STATUS (CUSTOMER-Closed Won), SALES REP, PARTNER, WEB ADDRESS, CATEGORY, DEFAULT ORDER PRIORITY, and COMMENTS.

Figure 7.1 – A portion of the Customer screen

Setting up the client's leads, prospects, and customers becomes second nature as you work with more than a few clients but keeping these rules in mind will help you avoid confusion or conflicts with other features (such as transactions) later on.

Next, we'll see how service businesses can use NetSuite projects to track service work as it occurs.

Implementing customer projects and resources

Any business that has people performing services, such as installations, repairs, training, or consulting, might want to use NetSuite's **project** features. Projects are directly tied to a customer record, and they include a lot of useful features allowing these businesses to track all the details that go into their project, such as the assigned resources and the estimated costs of delivering the work.

The first choice we make when implementing projects is whether the business has relatively simple requirements (and so might use what we call **basic projects**) or more complex needs, in which case they will use **project management**. For instance, if a company is primarily an electronics manufacturer, but they do perform field service calls on occasion, they might find it easier to work with basic projects. But if service work is their primary business, they will benefit from the more advanced project management features, such as task grouping and resource assignments, project templates, and cost accounting and reporting.

Just know that basic projects in NetSuite are *really* basic, and in practice, not many businesses that perform services will find them sufficient. See *Chapter 12* for more information on how to use projects within NetSuite.

Assuming you're working with a service business and have chosen to use NetSuite's Project Management features, you will want to become familiar with the main parts of a project, including the resources and task lists and the billing features. There's a lot to learn here, so it's most common to find consultants who specialize in this topic working with service businesses. Your first task is usually to help the client define their project templates (so that they can quickly create new projects with just the right sets of details and, in many cases, predefined tasks and sometimes resource types). With the templates set up, you can move on to training the users in managing projects and billing the services as they are performed. Services companies usually pay special attention to how their services are billed, so expect to talk to them about billing schedules or possibly charge-based billing quite a bit, as well as, many times, grouped invoices. We need to create customizations around these native features in many cases as well. As always, it's our job to ask a lot of questions to make sure we're gathering the right requirements at this stage.

New in the 2024 NetSuite releases is the **Field Service Management** SuiteApp. NetSuite acquired another company's product and has now integrated it into the ERP system. If your client's business needs help with scheduling resource visits, asset and inventory management, and billing for work done out in the field, this new module is well worth checking out.

Next, we'll talk about how to handle the sometimes complex connections and relationships between a customer's business and the people who work for each customer.

When to use contacts versus sub-customers

There are going to be times when you're working with a client to get through their customer-related requirements and how they think of their customers might be a little confusing. Some companies, for varied reasons, have very different ideas of what they call their *customers* versus those *customers'* *contacts*. Sometimes, the customer might be a company, and sometimes a person. Those people might work for companies, or they might be self-employed. Just keep the native NetSuite features in mind as your client explains how they think about these entities and you should be OK.

In NetSuite, it's very straightforward – a *contact* is someone you want to always be associated with a single lead, prospect, or customer record. Examples of contacts include the CEO, the billing manager, or shipping employees – really, it can be anyone who works for a customer's company and might need to be contacted via NetSuite. Contacts can be set up with access to **Customer Center** if that's active (see *Chapter 10* for more details on Customer Center), and they can have other custom properties associated with them. For instance, I've created customizations for clients where we added new checkbox fields to contacts and then used those to fine-tune which contacts received which email notifications coming from the system.

Sub-customers, on the other hand, are also *customer* records in NetSuite, so they have the same properties and actions as the parent customer does, except that they also have a link to their parent customer. An example where using a sub-customer makes sense would be the case of a franchise where you may receive payments from the parent company, but sales and invoices need to be managed separately for each franchisee location.. Using this feature is a good choice when there's more than just a separate address that differentiates the sub-customer. NetSuite remembers who the parent is and who the sub-customer is all the time and, generally, has some rules about how they're handled. These rules include the following:

- Invoices go out to the parent company unless you choose to change that
- Payments are expected to be received from the *parent customer*, again, unless you change how that is set up (see the **Help** page titled *Consolidated Payments* for more on this)

Deciding when to use contacts and when to use sub-customers should be fairly simple, then, assuming you can convince the client of the accuracy of your selected approach.

If the related entity is a person, and only their contact information differs from that of the customer company, then make them a *contact*. We want to keep this as simple as we can for the client and doing this achieves that goal. This is the case for the billing department within a company, for instance.

On the other hand, if the related entity has other things that make them different – such as a billing or shipping address, or some other primary categorization – then make them a *sub-customer* in NetSuite. This does make the setup a little more complicated since we will need to indicate this relationship when importing the customers list, but it's also more flexible. With a parent and a set of sub-customers, we can have the child company listed on a sale and the parent on the invoice, for instance. We can also send invoices to the child companies but then receive a consolidated payment from the parent. Lots of our clients will consider this a key factor in how they set up their customers, so it's really important that you (their consultant) know all the options NetSuite has to set up these entities.

Next, let's look at the other types of entities in NetSuite and what you need to know about your client's requirements at this stage.

Managing vendors, partners, and other entity types

NetSuite requires that we define a list of vendors to be associated with all the **procure-to-pay (PTP)** transactions. These are generally your clients' suppliers and other businesses they buy products and services from.

This list will include the client's sources for the products they sell, their favorite office supply store, their accountant, whoever they pay rent to, and so on. Vendors, like other entity types, can be a company or an individual. Some clients will confuse vendors with customers, so just keep in mind how NetSuite works as you work through this with them and explain the uses we'll have for this data once it's stored in the system.

For vendors, we need to know things such as their names and addresses, and we usually categorize them in some way meaningful to the client, and so on. Here's a sample **Vendor** screen in NetSuite for reference:

Vendor 🔍

Save ▼ | Cancel | Make Payment

Primary Information

VENDOR ID ★
Copied From Name ☒ AUTO

CATEGORY

COMMENTS

TYPE
☐ COMPANY
☒ INDIVIDUAL

MR./MS...

NAME ★

COMPANY NAME

JOB TITLE

Email | Phone | Address

EMAIL

ALT. PHONE

ADDRESS Map

ALT. EMAIL

MOBILE PHONE

PHONE

HOME PHONE

FAX

Figure 7.2 – The NetSuite screen for creating a new vendor

Talk to your client about how their vendors line up with the company's own subsidiaries. NetSuite allows one vendor to be associated with more than one subsidiary, but we would need to know now how that should work across the full set of vendors your client has.

It's also important to gather the requirements around vendor addresses now. In most cases, of course, companies have one main address for billing, and possibly many shipping locations (which can be important for returns, for instance).

But I've seen cases where a company's vendors are not that simple, and we get into complicated discussions about how to track their addresses since those affect how payments will be applied to bills later.

Since some companies have external people providing services for them, we need to talk about whether their vendors are going to be resources on their projects. For instance, a contractor might be added to projects to provide consulting. That can create complications again, but NetSuite should generally support this out of the box.

In addition to vendors, the system also allows us to track the following other types of entities, if we choose to (although this is much less common):

- **Partners:** Generally useful for any other entity (company or person) that might work with us but doesn't require a transaction with their name at the top (that is, not a customer and not a vendor). Talk to the client about whether they really need to track these in NetSuite, maybe in conjunction with promotions or because they need to give their partners access to NetSuite via **Partner Center**. (See *Chapter 10* for more on this center.)
- **Competitors:** It's the same idea here, except, of course, this time we will track other companies who sell products or services like your clients. Many of my clients considered this list very confidential; however, when we used this feature, it was mostly restricted to select management types. Of course, this is up to your client, though, so ask them how they want this to work.

Understanding a client's needs around these entities gets us closer to being able to start configuring the system. But we also need a solid idea of how the client thinks about their items, how they buy and sell them, and how they use them for miscellaneous purposes.

Identifying items and their types

Items are anything the client wants to purchase, sell, or resell, so that can mean a widget, a downloadable book, a service they purchase or provide, or any miscellaneous fee they add to sales. NetSuite supports many different item types for this reason. It's important to remember this when you get into configuring a list of items and their types in the account for your client. Each item type's screen has differing lists of fields, and each type of item has a unique GL impact on business reporting, so the requirement-gathering phase is a good time to help the client understand what they'll need to tell you about all of their items.

Most of my clients have no trouble with getting this list together at this stage, but they usually take a much narrower view of what's needed versus the reality. For example, they have a list of products they sell, but they don't realize at first that we also need to know what the various units of measurement are for each item (do they sell them as *eaches*, *cases*, or in *pallets*?). What will the default unit of measurement be for purchases and sales? What about other non-inventory items they might sell, such as printed user manuals or digital files? Are there fees they need to access in some cases, such as extra freight charges, late fees, or finance charges? Do they have services or other charges they need to add to orders and invoices? Or maybe they will occasionally want to see a descriptive line inserted with helpful comments.

I sometimes use the **New | Item** list in the system as a guide while talking to a client about all the different item types that NetSuite supports, and you can too, as long as you don't get too bogged down at this stage in the system's details. Here's what that looks like by default in the system:

New Item	
ITEM TYPE	MATRIX ITEM ASSISTANT
Assembly/Bill of Materials	Matrix Item Assistant
Lot Numbered	Matrix Item Assistant
Serialized	Matrix Item Assistant
Description	
Discount	
Inventory Item	Matrix Item Assistant
Lot Numbered	Matrix Item Assistant
Serialized	Matrix Item Assistant
Item Group	
Kit/Package	
Markup	
Non-Inventory Item	
For Purchase	Matrix Item Assistant
For Resale	Matrix Item Assistant
For Sale	Matrix Item Assistant
Other Charge	
For Purchase	Matrix Item Assistant
For Resale	Matrix Item Assistant
For Sale	Matrix Item Assistant
Payment	
Service	
For Purchase	Matrix Item Assistant
For Resale	Matrix Item Assistant
For Sale	Matrix Item Assistant
Subtotal	

Figure 7.3 – The New Item type selection screen in NetSuite

Most of the time, most clients will need a lot of inventory, non-inventory, and service types of items. For those, we ask questions about where they come from, where they're stored, and where/how they're sold. Are some for purchase only, some for sale only, and some for resale? ("Purchase" items can only be included in purchase transactions, "sale" items can only be included in sales transactions, and "resale" items can be included in any transaction.) These questions will guide you to help the client set up their item import lists later since we almost always get items into the account via **CSV** imports.

Then, we sometimes need to talk about more complicated item types as well, such as **assemblies**, **kits**, and **groups**. Businesses with some facet of manufacturing need *assembly*-type items, since they purchase smaller parts and assemble them into finished goods. It's good to drill into this process with your client since there can be complexities hidden in there if you don't ask enough questions. For instance, it's not uncommon for a company to manufacture some items in-house and to farm out that work to a partner as well. While *kit*- and *group*-type items also have a list of components they're made from, NetSuite places restrictions on us for how they can be bought and sold via transactions, so it's good to discuss any case where the client uses a term like *bundles*, as they can be handled via any of these item types. This is another place where making the wrong call now about what we'll use can cause real problems down the line, so if your client needs to use assemblies, kits, or item groups, you must know the differences and make the best choice for your client.

There are a few other item types to know about, of course, so let's have a look at those next.

When we use special-purpose items

You can see from the list in the previous section that there are a lot of item types in each account, and a business will largely control which of those we use. However, it's not at all uncommon for a company to need many types, so let's cover a few of the important use cases here (the rest you can just read up on the *Item Types* Help page and then be prepared to talk about when needed).

Lot numbered items

Any time a client needs to track items in sets, as they're received and when they're fulfilled (heading out the door), we assign lot numbers to those sets. A *lot* can have one specific item number in it or many, but the point is to assign these numbers to the items while your client has them in a warehouse. This is most common for food and beverage-type businesses, but anyone might have a use for this feature.

Serial numbered items

This is not a type unto itself, but a feature that you can choose to enable within the inventory and non-inventory types. Use this only when needed, for example, for things such as expensive electronics that arrive with a serial number from the manufacturer.

Matrix items

When any one item comes in a variety of colors, styles, and sizes, we usually end up using *matrix* items to keep track of all those options. This feature is very powerful, but it's also relatively complicated to set up and to use operationally, so explore whether you can make use of the **Item Options** feature instead before you commit to using matrix items. See the Help page in NetSuite for more on *Item Options*.

Other charges

The other charges item type is available so that you can set up things such as a subscription fee for a software company, or anything that will be part of a renewable contract. Think about how you sign up for a service online, when a company might need to charge you varying amounts depending on how much of their service you use each month. These variable quantity/rate items are easy to set up when you use other charges.

As we've said in previous chapters, gathering these and all of the related requirements for your clients is so important. By working with many different clients, you learn to ask the right questions over time, and you need to get all of their item requirements written down correctly now just the same. Hopefully, this section has given you a lot of guidance on doing that.

As you start to talk to a client about their list of items, if they will use any sort of **warehouse management system (WMS)**, then you need to make sure all of the choices you make for their items will sync well with that system. NetSuite has a WMS product that is tightly integrated with the ERP, and many other WMS applications can integrate with NetSuite too. How they do that exactly will affect how you should gather information from them about their items lists. For instance, if the WMS uses bins to keep track of where items are stored in the warehouse, you will need to enable the **Use Bins** setting on your inventory items in NetSuite. Some WMSs support license plate numbers for case tracking but NetSuite doesn't have that out of the box, so make plans now for how you will handle that. Likewise, it's important to know if the client plans to use the native NetSuite Inventory Count feature, Smart Count with the NetSuite WMS, or another inventory counting feature native to other systems. In other words, you will need to ask questions like this when you hear the client will be using any WMS.

Likewise, many clients with inventory items will use some form of drop ship or special-order items for at least some of the time. Drop shipping allows a company to order products from their suppliers, and then have them shipped directly to their customers, without involving their warehouse in the shipment. Some businesses run entirely with drop ship items, while others don't at all. Special order items are similar to drop-shipped items but are received into your client's inventory from the supplier. Natively, any inventory or non-inventory item can be set up for drop shipment or special order from a sales order, manually or automatically.

Using the right item type will help streamline your clients' operations and will make them happier with NetSuite in the long run. Learn the differences between all of the available types early in your NetSuite learning so you can assist clients with this, with confidence.

Summary

In this chapter, you learned how to start talking to your client about who their customers and other related entities are, and what their items list will consist of. When you talk to clients about leads versus customers, just remember they might use different terms, but you can help them translate their terms into NetSuite features in all cases. We also learned how service-oriented businesses can use projects to plan for and manage their employees' work. When you gather the client's vendor, partner, and competitor requirements, focus on how the data will be used within the system to make sure things stay on track.

When it comes to your client's items, talk initially about what they're used for, and how they're purchased, sold, and distributed. This will be your guide to how they should be set up in NetSuite, including their type and the other important features you'll need to enable for each.

In the next chapter, we'll learn how to collect the requirements around the client's business processes, which will lead to the creation of transactions in the system. Usually, there's a lot to unpack with that topic, but we can break it down into manageable chunks and get the work done. You'll see how we do that too.

Self-assessment

Take a moment now to think back over this chapter's contents and test your understanding of the topics covered here. Think about how you would apply the following concepts in your day-to-day work:

1. How would you respond to a client who tells you that sometimes they buy things from another company, but they also sell things to that same company, and they want to have them set up just once within their account?
2. Acme Electrical Services has a team of installers and repair technicians they send out to other sites to install and repair equipment. They typically sell these services as a combination of subscription offerings and one-off service tickets at their customers' request. How can you help them set up project templates in NetSuite to make this process as efficient as possible?
3. Your client lets you know that they have one company they consider a customer, which has over 2,000 people working within it, and they want to enter/track all of those people in NetSuite. They need to be able to sell items to any of those people but bill them to the company. How would you recommend they set up this company?
4. A client has individuals they work with who are employees of multiple businesses, all of whom might be their customers. Is there a way to set up these contacts in NetSuite so that they can be associated with more than one customer?
5. A business sells furniture, and many of its items come in a variety of colors and have multiple fabric options. Your client has signed up for a SuiteCommerce e-commerce site, and they want to make sure shoppers can select from these options easily. Which item type is right for this situation?
6. How would you handle the situation of a customer saying they need to have 2 million items in their inventory? They don't actually sell all of those items but say they might and so want to be able to work with any of those items as soon as they need them.

Identifying the Organization's Main Transactions

NetSuite transactions are the heartbeat of the system. We use them to record a business's future plans and daily activities. Most NetSuite clients use a mix of standard transactions, such as purchase orders and vendor bills, and less common transactions such as assembly builds, item transfers, and revenue arrangements. Understanding what NetSuite offers and helping your clients determine how they're going to use transactions correctly is your next task.

In this chapter, we will cover the following topics:

- Grouping records by business process
- Gathering requirements for **record-to-report (RTR)** transactions
- Gathering requirements for **procure-to-pay (PTP)** transactions
- Gathering requirements for **order-to-cash (OTC)** transactions
- Analyzing the other process groups

When performing the steps in this chapter with a client, you will help the subject-matter experts for all the teams who will interact with transaction screens, including the finance and accounting teams, understand how they will use NetSuite daily. From *Chapter 13* to *Chapter 16*, we dig into using these transactions in NetSuite.

Grouping records by business process

There are many ways we could break down the list of activities a business needs to record each day they're in operation. The most common method that's used in most industries today is to label them by the process flow they are a part of, describing the starting point and the result. For instance, for sales to customers, we refer to the OTC process, since the process starts with an order and ends with the business being paid (hopefully!). We use these labels to describe the processes almost every business has, no matter which industry they're a part of or where they are in the world.

As we progress through this chapter, please note that sometimes we'll use the word **record** for things you can enter in NetSuite. These are usually going to be transaction records but may on occasion be something else, such as an item or a support case. We don't want to get too bogged down by those terms here, so just remember that every screen in NetSuite where we enter some data and then save it creates a record.

When you're gathering these requirements with your client, it may not be necessary for you to explain all the nuances of these process names to your client, but it will help for all of you to have this common language to use and a basic understanding since you'll need to be able to split up the work and talk to each component separately (and may have different people working on each process with the client).

Here is the most common list of business processes we use today and the ones we'll reference frequently throughout the remainder of this book:

This Process	Is Used For	Has These Associated Records
Design to Build (DTB)	The basic setup of items and related details used within transactions	Items (all types), item costing, and pricing. (If applicable) work orders, assembly builds, and more.
Call to Resolution (CTR)	Tracking interactions with customers and others	Cases, issues, solutions, and knowledge base.
Record to Report (RTR)	Defining the way we want accounting and other related processes to work in the system	Accounting periods, journals and other financial transactions, budgets, and bank reconciliation.
Hire to Pay (HTP)	Tracking employees from recruiting to hiring to the end of their employment	Employees, payroll, and others as needed.
Lead to Quote	Tracking potential customers and the deals we might make with them	Leads, prospects, customer-specific pricing, and opportunities.
Marketing to Return on Investment (MTR)	Defining advertising and other marketing efforts, and evaluating their success	Campaigns, emails, and marketing reports.
Order to Cash (OTC)	Tracking estimates and sales, fulfillments, and shipping, through to billing and payments	Quotes, sales orders, cash sales, item fulfillments, and invoices.

This Process	Is Used For	Has These Associated Records
Procure to Pay (PTP)	Tracking the company's purchases, expenses, receipts, bills, and vendor payments	Purchase requisitions, purchase orders, expense reports, item receipts, inbound shipments, and vendor bills.
Return to Credit (RTC)	Tracking vendor and customer returns, receipts, credits, and refunds	Vendor returns, return authorizations, vendor credits, and credit memos.
Project to Delivery (PTD)	Tracking service delivery, from defining the project to confirming all work is completed	Projects (also known as jobs) and resources.
Web to Order (WTO)	Defining the e-commerce presence, from domains to the web store and all of its functions for completing orders	Websites, domains, item categories, and other records used by the client's web store.

Table 8.1 – The business process names and the names of the related records in NetSuite

These are the business process names we'll use from now on, and as you can see, each process maps to the names of many records and transactions in the system. They're meant to be shortcuts to save time in conversations but please also remember that they're not set in stone – and not every NetSuite client will use all of these processes.

Next, let's start to drill into the main business processes we always need to gather requirements for, starting with RTR.

Gathering requirements for RTR transactions

The name of this process sounds very generic, but this is for a good reason – it is. You may be thinking *RTR could include anything*, but it's meant to convey a sense of the general setup we need to do in the account and then a financial connection. This is a process every business (including non-profits) has since it includes things such as setting up the accounting periods and accounting preferences and defines how they will use journal entries and perform period closes. This also includes any other sort of around-the-edge adjustments that every business needs from time to time. Whether they sell inventory, collect donations for a charity, or trade cyber currencies, every business needs to make occasional adjustments to their accounting.

Here are some topics to cover when analyzing the client's RTR needs:

- How do they want to track accounting periods in NetSuite? (We talked about this in the previous chapter in depth, but this is also usually included in the RTR discussion.)
- How does the business record taxes? Get the full list of agencies they must report and pay taxes to.
- Are there any common, recurring needs for RTR transactions? For instance, does your client eliminate account funds once a month as part of the closing process period?

You'll primarily work with your client's accounting and finance teams for the RTR processes.

Next, let's take a deeper dive into the client's purchasing and **accounts payable (A/P)** billing processes.

Gathering requirements for PTP transactions

Within this process, we cover the purchasing needed to make the company run. If your client is a software company, their PTP transactions may be limited to things such as rent, utilities, and computer and office supplies. In that case, their requirements for these transactions should be simple to gather. But many companies have rules they've established around things such as how their employees should make requests for purchases, or which management approvals are needed when these employee requests come in. Approvals are very common for PTP transactions, even when the company has dedicated buyers.

Here are some topics to cover when analyzing the client's PTP needs:

- We need to understand the process flow within the business for the entire PTP process, which will include purchases, receipts, bills (also known as vendor invoices), and returns and credits.
- Is there any direct communication between the business and the vendors they purchase from? For example, do they need to allow their vendors to log in to the NetSuite Vendor Center for any reason? Or do they receive bills and shipping notices from vendors via fax, email, or electronically?
- Which transactions will require an approval process and exactly how do those approvals need to work? And who is allowed to edit them, for what reasons, and at which stage in the process? Many times, we can solve these requirements with a simple workflow, but some companies have more complex requirements and levels of approval needed. Now is the time to fully understand how they want this to work and to recommend a simpler approach, if possible.
- How will the client handle the receipt of goods they purchase? Specifically, will they ever need special handling after the receipt (such as splitting up a case into smaller units for sale)? Or will they need to integrate their receipts with any warehouse management service/application?
- Do they ever need to handle vendor prepayments or rebates?
- Do they need inter-company transactions, where one subsidiary purchases items, and then those are transferred to or sold by another subsidiary within your client's business?

As you work with your client's purchasing people to understand their needs, remember what we said earlier about how they're probably not savvy with NetSuite's features yet, so do your best to work with their terms for now, translating those into NetSuite terms over time.

Next, let's see what information you need to gather from the client's OTC people.

Gathering requirements for OTC transactions

When it comes to sales tracking, businesses tend to fall into a few known categories. If they have a retail component or only sell items via the internet, they will most likely use cash sales to record them in the system, assuming all orders are paid for at the time they are placed. If they allow sales based on terms (Net 30, for instance), then they'll need invoices, and so on. Spend a good amount of time talking to the sales and other related teams to understand their OTC process thoroughly.

Here are some topics to cover when analyzing the client's OTC needs:

- Do they need to record quotes, or estimates, in advance of a sale? Some businesses will use these as a way to start the sales conversation with their customers or to establish pricing in advance.
- What forms of payment do they accept for sales and when do they offer terms as an alternative? Do they accept customer deposits in advance of sales?
- Are there ever restrictions on which items each customer is allowed to buy?
- How do they handle pricing on sales? Some of my clients have had pretty complex rules around pricing, making them customer-specific and item-specific, with different prices in effect for varying date ranges. NetSuite has a couple of solutions that can help with situations like this, and then there's always the possibility of creating custom solutions when necessary.
- Do they need to offer rebates of any sort to their customers?
- Will some items or services be sold via contracts or subscriptions? This is something you should know before you start working with a client, as an output from the sales cycle, but it can occasionally pop up late in the process too.
- Are any items sold via dropships? This is usually easy enough to handle, but it's still important to know when you need to cover this functionality.
- Does your client have any items they rent? For example, college bookstores can rent textbooks to students, directly or via a third-party service. A storage company rents lockers to its customers. Accounting for rent is a special case in terms of subscriptions since they can be very short-term.
- How are sales taxed in the places where your client sells? If you're in the US, you might be able to solve these requirements just by installing and configuring a partner bundle since, fortunately, all businesses must follow the same local/county/state/federal government rules. Outside the US, things get a little more complicated, so it's always important to have someone on your team who knows the rules for each locale where your client sells items or services.

- Are approvals required for any sales transaction? Who is allowed to edit them, for what reasons, and at which stage in the process? For instance, some businesses will let anyone edit any sales order, at any stage in the process, and then some place restrictions on that kind of editing once the order reaches (for instance) the pending fulfillment stage. Just be mindful that overly complicated workflows can slow down the system, so performance is always a concern.
- How do they ship the products they sell? Are integrations needed with any third-party logistics service or warehouse management application?

Work with the sales, customer support, or whichever department processes sales and handles billing to understand the *five Ws* we mentioned in *Chapter 3*; for instance, *who* does the work (are there sales reps and sales managers with different permissions, for instance?), *where* do they do it (on the phone, in the field, from multiple offices, and so on), and *when* and *why* do they make changes to orders? For most of my implementations, we've spent the majority of our time getting the OTC process nailed down, making sure *what* they enter and how they do that works exactly as required by the client since that's their customer-facing part of the business and it must work perfectly.

In 2022, NetSuite announced what they call the **Rebates and Trade Promotions** features. These are especially useful for companies that receive rebates from their suppliers and/or offer rebates to their customers. NetSuite offers a rebate management SuiteApp, which incorporates both what it terms **vendor rebates** and **customer rebates**. These can be defined so that they are then automatically or manually applied to transactions (purchases and/or sales) and even the month-end or other frequency processing of the rebate amounts can be automated. Many industries in the US and elsewhere use rebates to make their brand or company competitive in the marketplace, so having NetSuite automation manage them is a key selling point. Make sure you familiarize yourself with these features, especially if you work with clients in the wholesale/distribution industry.

There are many more processes to understand and for you to at least briefly talk about with your clients, so let's explore some of those in the next section.

Analyzing the other process groups

If you review *Table 8.1*, you'll see that the other remaining business processes are going to be relatively more or less important to each of your clients, based on their industry and their specialty within it. Not every NetSuite client uses the *cases/issues/knowledge base* features, for instance, but those that make them a part of their customer-facing business consider them to be very important indeed.

Let's talk through a few of the more commonly used processes here and get an idea of the information you need to gather from the client when analyzing them.

Design to build

In the previous chapter, we talked about gathering requirements for items, but this process includes more than that. If your client has a manufacturing department of any sort, you might need to understand their need for work orders and assembly builds or more transactions, depending on how deeply they wish to track their shop processes.

NetSuite's **Advanced Manufacturing** features include the ability to track all work in progress, step by step, for instance, if that's needed.

Lead to quote

Some companies have to work hard to earn new customers, following a daily process for converting leads into prospects. We track this conversion with the transactions in the **Lead to Quote: Opportunities and Estimates (or Quotes)** functionality. When you work with one of these clients, be prepared to talk to them about things such as how they can check the status of conversions, and how to review/approve their opportunities and quotes. Commissions for salespeople also come up here, since converting a quote into a sale is a big deal at these companies. Check out the NetSuite **Help** pages under *Commissions* for how the system calculates commissions and confirm with the client whether this could work for their teams.

Call to resolution

For companies with strong customer support departments, NetSuite's case and issue records can be very useful. Talk to the client about who these people are, what types of issues they deal with, and how those things need to tie into their OTC transactions, such as sales orders and return authorization cases. Having issues and orders closely tied together can be a key differentiator for software and other types of businesses.

Hire to pay

For most of the clients I have implemented, we set up employees, import their list, restrict permissions to the right roles/people, and we're done with this process. But for companies who have more requirements around things such as payroll, sales commissions, or recruiting, we need to dig deeper to understand how they perform these activities. When necessary, we can add the SuitePeople package, but it's also common to bring in a set of third-party SuiteApps or bundles to add to NetSuite's native functionality in these cases.

Marketing to return on investment

When you're working with a company that has a one-person marketing team, there's not a lot to set up in NetSuite here. But when the company does a lot of marketing in-house, you need to understand things such as how they handle their campaigns, upselling, and possibly promotions, depending on which team owns that within the company.

Revenue recognition

While this topic is really part of OTC, it's good to call it out here as a special case you may need to deal with, depending on the client's business. When revenue recognition (or *rev rec*) is important to the business, you'll want to be sure you have someone dedicated to gathering these requirements thoroughly since this can have a huge impact on the business's bottom line. This typically applies to software and some services companies, but any business can have one product or service that requires this special accounting. Check out *Advanced Revenue Management* in the Help pages for more info.

Return to credit

We're calling this out separately from PTP and OTC because some companies have much more going on with returns than the average business. For businesses such as clothing retailers, allowing returns is a big part of the business, so we can't skimp on defining the processes they follow. In these cases, there are typically lots of rules to be implemented in NetSuite, including how many returns a customer can make in a defined period, or exactly how each item is to be processed when received as part of a return. This can be complicated by things such as rebates and discounts as well since the amount to be returned has to take those things into account.

Project to delivery

Businesses that provide services (such as deliveries or consulting) will typically either use NetSuite's **Project** feature, subscribe to OpenAir if they're large enough, or have chosen another separate project management tool, outside of NetSuite in some cases. If they're using NetSuite for this, then you should come to understand how they create new projects, how they staff them, and what they need to keep track of. For many businesses, cost and time are the two main metrics, but we can also set up tracking for other things as required. This is yet another case where having someone who specializes in services work and also knows all of the related NetSuite features well, really makes a difference in the success of an implementation.

Web to order

For clients signing up to use NetSuite's SuiteCommerce or SuiteCommerce Advanced features, you need a team of e-commerce specialists to analyze their requirements and to build them out into the software. This can include setting up multiple websites, domains, and all the other related features, and we very commonly receive at least a few customization requests from these clients. When a company sells via the web, they want that user experience to be very easy and frictionless, so getting this right takes a lot of experience. Don't try to take on an e-commerce implementation by yourself, without having someone more experienced to guide you, in other words.

These processes are the main ones covered by features within NetSuite, but your client may have others that are just as critical to their business. As always, whenever you talk to them about anything, use your active listening skills to try to pick up on any mention of something new/different they're going to need, and always spend at least a few minutes exploring anything that sounds interesting.

As you can see, there's potential here for needing to have a lot of conversations with each client, but don't feel overwhelmed. You can do it, and you will succeed with your clients if you learn to break down every problem you encounter into manageable tasks and then tackle them one at a time. The larger your implementation team is, the more you can divide up this work. Your team's size should scale with the size and complexity of your clients.

Summary

In this chapter, we started by understanding your client's business processes by breaking them down into categories (PTP, OTC, and so on). Then, within each category, we got to know how the business functions – who does the work and how they do it. It's at this stage in every implementation project that we get to understand what makes our clients unique, and this allows us to tailor the system to their exact needs.

Once you've completed a few NetSuite implementations for a variety of clients, you start to think you've seen it all, but don't fall into that trap – every business has at least a few elements that make them unique and it's part of your job to make sure you hear your client's description of their processes fully, and then to ensure those elements are mapped into NetSuite successfully. You have to understand their processes first so that you can move on to the next phase, which is where we start to configure the system to meet all of the client's requirements.

In the next chapter, we will move on from gathering requirements to configuring the NetSuite account to meet the client's needs. We will learn how to use SuiteBuilder's various features to define custom records, fields, and forms. (We will finally get to do some hands-on work!)

Self-assessment

Take a moment now to think back over this chapter's content and test your understanding of the topics covered here. Think about how you would apply the following concepts to your day-to-day work:

1. As you start working with a client, they keep referring to their main sales transaction as a "Commitment". How you can you help them translate that into an appropriate NetSuite term?
2. The senior accountant for your client lets you know that, occasionally, they will personally need to enter journals into the system that no other employee should be able to view. What should your response be, or how can you help meet this requirement?
3. Your client's business has a high sales volume, with around 5,000 new orders placed on an average day. What special questions should you ask in cases like this? (And how should this affect your designs for setting up their orders in the system?)

4. A business uses quotes to work out deals with their customers, and they will need to be able to combine multiple quotes into one order regularly. Is there a native feature in NetSuite that can help with this, or how could you meet this requirement?
5. Your services company client wants to be sure that every project is directly tied to any related sales orders. What's the best way to ensure this happens in their NetSuite account?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search for this book by name.

Note: Keep your purchase invoice ready before you start.



Part 3:

Implementing an Organization in NetSuite

Now that you understand more about NetSuite implementation and your client, you will learn how to put all that into practice by configuring their account and testing everything with the users. New in NetSuite 2024/25, you'll find new information on AI and many other new features, as we will cover in this part.

This part has the following chapters:

- *Chapter 9, Custom Forms, Records, and Fields*
- *Chapter 10, Centers and Dashboards*
- *Chapter 11, Items and Related Lists*
- *Chapter 12, Customers, Vendors, Contacts, and Other Entities*
- *Chapter 13, Financial Transactions and Period Closes*
- *Chapter 14, Procure-to-Pay Transactions*
- *Chapter 15, Order-to-Cash Transactions*
- *Chapter 16, Other Transactions and Custom Transactions*
- *Chapter 17, Analytics, Reports, and Data Exports*

Custom Forms, Records, and Fields

An ERP system would be unusable without the ability to customize screens, allowing people to add the fields they need for their business and remove all of the natively offered things they won't use. Every NetSuite client will have a few (if not many) reasons to customize their NetSuite instance, and that usually starts with the records, forms, fields, and other related features. You will learn how to use these always-available custom fields, records, and forms to tweak the account to fulfill your client's requirements.

You will be working with the account administrators and the SMEs for each department in the business to configure the forms and records they will use within NetSuite for both native and custom data types.

In this chapter, you will learn how to implement a client's foundational elements, including the following:

- What is **SuiteBuilder** and when should you use it?
- Customizing entry and transaction forms
- Using custom record types to store additional non-standard data
- Defining custom lists and fields
- Change management for all your custom objects

By the end of this chapter, you should have gained the skills to walk a client through the initial setup of their account, including setting up the main forms they will use every day, plus the basics of the custom records and fields they need to make the account specific to their needs.

To do well in this chapter, you should know the basics of getting around in NetSuite and know how to adjust the system's out-of-the-box screens with custom features. If you've read *Chapter 2* on implementation methodologies, you should know that the process we describe here is most like the Waterfall method, which I said makes sense for an implementation project for a new NetSuite client.

What is SuiteBuilder and when should you use it?

When we think about NetSuite, one of the first things that should come to mind is how flexible and configurable it is for any ERP-related use case. Most of that configuration is accomplished via the SuiteBuilder parts of the system.

SuiteBuilder is not a screen you go to though; you will not find it in the menus across the top of the screen (in NetSuite, these are called **centers**). Instead, you will find the various SuiteBuilder features under the **Setup** and **Customization** menus, allowing you to rework all of the native record screens NetSuite offers and more. When you need to change the order of fields on a screen or add a new field to one, you will use the SuiteBuilder features to accomplish this. In older ERP systems, you might have to hire a professional to make changes like this, but in NetSuite, any user with some training (and the right permissions) can make modifications to screens and records (and more) to suit the business’s needs. As the business changes, the system can be updated to match – again, without any coding or other technical experience necessary.

Here’s a list of all of the features that fall under the SuiteBuilder umbrella and when you might want to customize them. We will cover the most important topics in greater detail in the rest of this chapter:

Feature	When to use it
Forms	When you want to change the fields, lists, and tabs on any native record screen
Fields	When you want to add fields to a native or custom record screen
Record types	When you need to store an entirely new table of data in your NetSuite account
Transaction types	When you want to substitute a new, highly customized version of a native transaction, such as a sales order or purchase order (rarely used, but very important when required)
Segments	When you need to track segments beyond the location, class, and department
Templates	When you want to customize how a form will be output from the system, on an email, PDF, or HTML page
Centers	When you want to add new custom menus, sections, and links to the native ones across the top of every NetSuite screen
Subtabs	When you want to add custom tabs to a form to visually separate a group of fields or new lists
Sublists	When you want to add custom lists to a form or a subtab with data from another native or custom record, or a set of search results

Table 9.1 – List of SuiteBuilder features in NetSuite

Depending on your client's requirements, you may not need to customize all of these features for every implementation, but knowing how to do this, and how your changes can affect the user's experience working with the system, is an especially important skill for a NetSuite consultant or administrator. Another important aspect of this is knowing how and when you can undo your changes. For instance, if you enable a feature such as **Advanced Inventory** and then customize your forms to use that feature's new fields, you need to know what will happen to your forms if you decide to disable the feature again sometime later.

As you start to customize the features within a client's account, always remember that you might need to undo a change or customization at some point (if the client changes their mind, for instance). Keep track of the changes you make and be prepared to remove something or deactivate/disable it if the need arises. Know what other changes you might need to make each time you have to do that, too.

Side note – new Redwood theme!

Even though this is not specifically about SuiteBuilder, I want to mention the new UI theme NetSuite made available within the 2024.2 version. It's called **Redwood** and it's a major change from the previous default UI. Redwood is actually part of a major push from NetSuite's parent, Oracle, to improve the look, feel, and function of all of its software. More information is available about this at <https://redwood.oracle.com/>. As of this release, NetSuite users can choose to enable the theme via their user preferences (**Home | Set Preferences**). When they do this, all of the NetSuite screens will follow this new design pattern and also gain interesting new features. Go ahead and experiment with this and see whether you like the new direction!

Let's dig into the SuiteBuilder features then, starting from the top (forms) and working our way down (to records, fields, and more).

Customizing entry and transaction forms

When we visit a screen that is used to display any one record in the system, such as **Customers** or **Invoices**, we are looking at a *form*. NetSuite uses that term to refer to the customizable aspects of a screen. So, for instance, when we look at an inventory item, we see the screen arranged in a certain way, with groups of fields in sections, and with subtabs holding more fields and lists of sub-records, too. We see sections labeled as **Primary Information** and **Classification**, and here we see subtabs for **Purchasing** versus **Sales/Pricing** information and many more.

Inventory Item

Save

Cancel

Primary Information

ITEM NAME/NUMBER

UPC CODE

DISPLAY NAME/CODE

VENDOR NAME/CODE

PRIMARY UNITS TYPE

PRIMARY STOCK UNIT

PRIMARY PURCHASE UNIT

PRIMARY SALE UNIT

PRIMARY CONSUMPTION UNIT

PRIMARY BASE UNIT

PRODUCT NAME

SUBITEM OF

Classification

SUBSIDIARY

Parent Company

Parent Company : PRES Sporting Goods

☐ INCLUDE CHILDREN

DEPARTMENT

CLASS

LOCATION

Purchasing/Inventory

Sales / Pricing

Accounting

Revenue Recognition / Amortization

Web Store

Communication

Preferences

System Information

Custom

Item/Cost Detail

☐ TRACK LANDED COST

COSTING METHOD

Average

PURCHASE DESCRIPTION

STOCK DESCRIPTION

☐ DROP SHIP ITEM

Figure 9.1 – An example of a form, the new Inventory Item screen

Most of these visual features are customizable via a form. NetSuite stores many forms for each record and allows us to choose which users see which forms by default. It also keeps separate forms for each item type and each entity, transaction, and so on. We could work with just the standard forms that come with the system out of the box, but in almost every case, we create a custom version of the standard form and incorporate changes into that (we cannot ever change the standard forms; they're meant to stay there, as delivered, for reference if we need that).

We usually work to define a set of custom forms for every entity and transaction feature the client will use in their account, with the name of the form having their initials. So, if the business is named *ABC Corp*, the form's names might all start with *ABC*. If we need to create more than one form per transaction or entity, we want to be sure we're clear with the client on why that is; sometimes it's because we want to show different forms to certain user groups by role. Other times, it's because they have different use cases, such as with purchase orders; they might place direct orders some of the time, and drop ship orders the rest – that's a good reason to have two Purchase Order forms.

There are many reasons to customize the forms for a client, but they all boil down to the main ideas explained in the following sections.

Removing fields the users do not need to see or will not use

As you work to make the client's NetSuite account suit their business needs, it will always be beneficial to the users if you can remove unneeded fields from the screen. For instance, if they never have a customer-originated PO # on their sales, then remove the **ORDER #** field from their **Sales Order** form. Do this for as many fields as possible, with the goal being to clean up the screen and make it as simple as possible for the users.

Here is the default **Sales Order** screen in a new NetSuite account. You get the fields NetSuite thinks most businesses need, but you need to review them and decide whether your client needs each of these fields.

summary	
SUBTOTAL	0.00
DISCOUNT ITEM	0.00
TAX	
TOTAL	0.00

Figure 9.2 – The default Sales Order screen

Take the time to go through the fields on each form they will use and look for opportunities to remove fields wherever it makes sense.

Adding fields the users need that were not included out of the box

By the same logic, if the users need to track something that NetSuite does not include in its default screen for something such as customers or vendors, you can create a new custom field and place that on the form just where you want it to appear. We will talk more about handling custom fields in the next section but always talk to your clients about how each of these custom fields' values will be populated. As you do that, you will get lists of fields from them, and you can either add them all at once or over time as they come up.

It is fine if they populate fields manually, by the person filling in the form, but if the client prefers the field to get a value automatically, you will need a plan for making that happen. You can do that with workflows or scripts, or via other methods. As an overview, we will cover those things in *Chapter 18*, or you can search the **Help** screens for these topics to learn more.

Rearranging the fields and lists on the form

The system does allow us a lot of flexibility in how the forms are put together, so if you add new fields to the form, you can move them to any section or subtab you need. Grouping a set of related fields together either within a section (a **field grouping**) or on a new subtab can keep the fields together visually, which generally helps the users remember how to find those fields when they need to populate them.

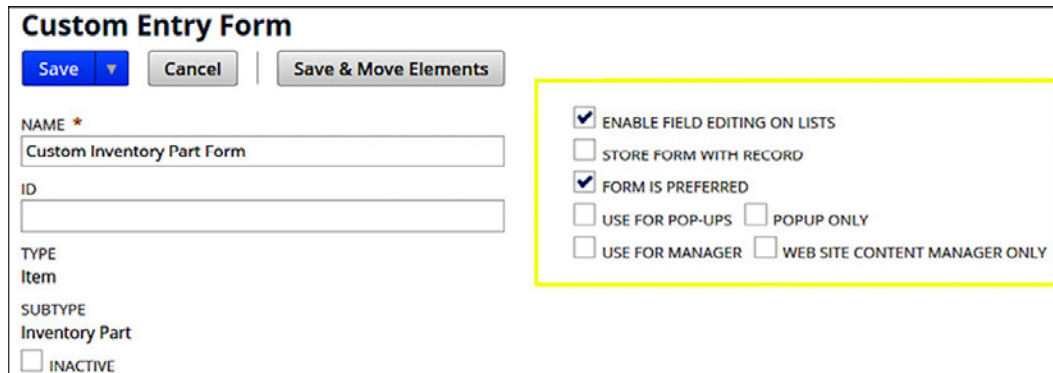
Every form has a **Main** section, and then you can add sections to every subtab and field group within those. Just try to restrain yourself and your client, with the aim being simplicity and ease of use. Without someone always pushing for these things, I have seen forms that became unusable in terms of extremely high complexity and lack of organization. It is OK to add things to forms as you think of them, as long as you make it someone's job (typically, not long before going live) to revisit every form and clean them up, move things into sensible groups/sections, and remove fields or lists that are no longer needed.

Along with these visual field settings, forms in NetSuite also allow us to customize other aspects of their use. We will study these in the following sections.

Enabling related features and settings

NetSuite allows us to make some basic choices for every form we customize. These settings will vary from one form type to the next. For instance, for items, you can decide whether the form should be used for popups; for transactions, you can decide whether to allow the use of the **ALLOW ADD MULTIPLE** buttons.

This is what the header fields of a custom form for an item look like in NetSuite:



Custom Entry Form

Save Cancel Save & Move Elements

NAME *
Custom Inventory Part Form

ID

TYPE
Item

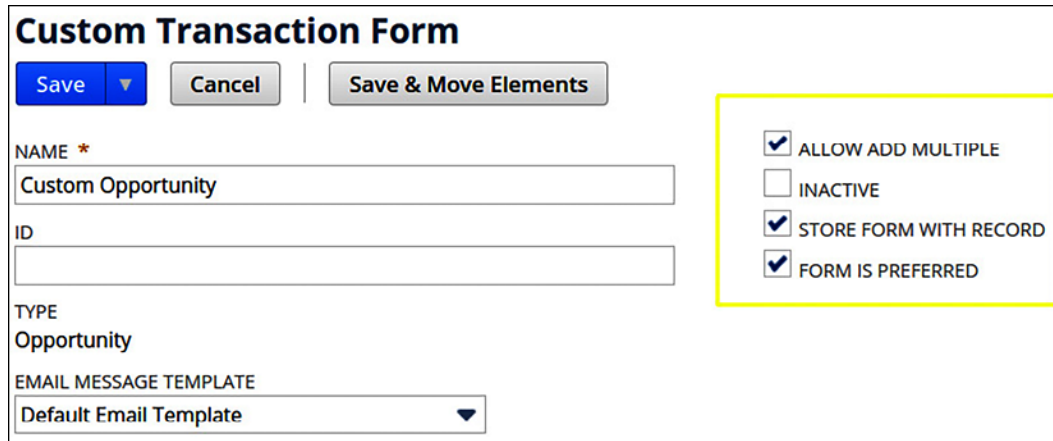
SUBTYPE
Inventory Part

☐ INACTIVE

☒ ENABLE FIELD EDITING ON LISTS
☐ STORE FORM WITH RECORD
☒ FORM IS PREFERRED
☐ USE FOR POP-UPS ☐ POPUP ONLY
☐ USE FOR MANAGER ☐ WEB SITE CONTENT MANAGER ONLY

Figure 9.3 – The Custom Entry Form screen for an item in NetSuite

And this is what the top of the screen looks like when creating a Custom Opportunity form:



Custom Transaction Form

Save Cancel Save & Move Elements

NAME *
Custom Opportunity

ID

TYPE
Opportunity

EMAIL MESSAGE TEMPLATE
Default Email Template

☒ ALLOW ADD MULTIPLE
☐ INACTIVE
☒ STORE FORM WITH RECORD
☒ FORM IS PREFERRED

Figure 9.4 – The Custom Transaction Form screen for an opportunity in NetSuite

Note the differences between these two figures. While every form has a **NAME** and an **ID**, items have checkboxes for popups, since items can appear within pop-up windows, but transactions do not. Item forms can also be set up specifically for managers and for use within a web store.

Storing the form with the record

This is useful when you have multiple forms for any record and want to ensure that all users see the same set of fields, lists, and so on that the user who created the record saw. With this setting enabled, we can ensure this, even when the other users are assigned to a different default form.

Actions

NetSuite allows us to add and remove actions available on a form. For instance, if you want to restrict most users from being able to print a customer record, you can remove that action from the form they are assigned to. You can also add new custom actions to a form via a script, but we generally use this only in rare cases (see *Chapter 18*, where we will cover this in greater detail).

Roles

This is one of those powerful features of the system that you can abuse and can then confuse people very quickly if you are not careful. It is great that you can have as many forms as you need per record and assign different forms to diverse groups of users via their roles but always keep this down to just a couple of options. If you find yourself setting up more than a handful of forms for one record, just know that people will get confused quickly, saying things such as “Why does Mary see the Categorization fields when she looks at these records, but I don’t?” As stated earlier, use restraint and consider other options before setting up anything too complex.

In this section, we learned how the various features of forms in NetSuite can be used to profoundly change the user’s experience when working with records. We can remove fields, add lists, and so much more. Experienced NetSuite consultants use these things to show users how adaptable NetSuite can be to their company’s needs.

Next, let us see what you can do when you need to store a whole new type of data in the system, with custom record types.

Using custom record types to store additional non-standard data

I have had many reasons to create custom records in the past. For instance, a client wants to keep track of warranties on a certain range of products they sell but NetSuite does not have a warranty record type. Any time we decide it’s appropriate, we can create a new custom record type consisting of fields, forms, and subtabs, and assign it specific permissions by role very quickly, all via the NetSuite point-and-click interface. The following is an example of such a custom record type, which you might use to track warranty data related to sales:

Warranty

Save Cancel

NAME *

☐ INACTIVE

WARRANTY ITEM

SERIAL NUMBER

SALES TRANSACTION

EXPIRATION DATE

EFFECTIVE DATE

Notes Files

User Notes

Remove all

TITLE	MEMO *	DATE	TIME
		4/19/2023	8:22 pm

Add Cancel Insert Remove

Save Cancel

Figure 9.5 – Example of what a Warranty custom record might look like

We can define just the fields we need on that record as well, of course. Using the warranty example in *Figure 9.5* again, we might add fields such as **WARRANTY ITEM**, **SERIAL NUMBER**, possibly a link to a sale, and a few dates we need to keep track of. If there can be multiple items linked to one warranty, we might want to use the very handy **Record is parent** feature, to link one field on the parent record type to a field on a related child custom record type. With this, the users can create one of the parent records, representing the warranty, and then as many of the child records as they need, representing the items covered under that warranty. NetSuite handles the joining or relating of those tables for us automatically, which makes using that data in things such as searches and reports easy.

As you create custom record types for your clients, always think about how many rows of data each record type will store over time. If the record type might have millions of records after some time (such as a year or two), then it is important to talk that over with the client. NetSuite does not charge extra for more rows of data in these tables, but the system's performance can suffer if there are often times when users need to search and filter or report on those results. If you allow a record type to end up with 20 million records and users need to export most of that list regularly, they will not be happy with how long that task takes. You need to take the time to understand the volume and scalability of each of these record types you set up and make sure the client has a plan for managing that data in the long term. In some cases, we can set up simple automation to delete any record that is more than 12 months old, for example, to keep these lists from growing without any restrictions. That requires a script to be developed, which is something we will cover more in *Chapter 18*.

Custom records are powerful and make a difference for many clients who need to store data that NetSuite did not anticipate. Over time, I am sure you will find many ways to use them to meet client requirements. Keep the maintenance of all that additional data in mind as you develop them.

Defining custom lists and fields

Many businesses will start a NetSuite implementation with the idea that you must help them make the system look just like their current ERP application. Doing that is not a good idea, since NetSuite works differently from most other ERP systems, but we can still accommodate the most important of the client's requests with features such as custom lists and fields. We can add new custom fields either to most native records or to our custom record types. Just be sure you only do that when there is no native field to serve a given purpose and ensure the client understands they will be responsible for populating the data for each new field. NetSuite offers a range of field types, from plain text (in various lengths) to lists, numbers, dates, passwords, and more. Review the Help page called *Field Type Descriptions for Custom Fields* to see the full list and keep most of these in your head at all times so you'll know how you can solve any problem that comes your way.

For instance, if a client says they want to add a **Standard Discount** field to the sales order, know whether they should use a decimal number or a percentage field. If they want to add a **Customer Service Comments** field to each **Customer** screen, know the difference between **Free-Form Text**, **Text Area**, **Long Text**, and **Rich Text** field types, so you can help them choose the right option for their needs. The difference is the number of characters supported in each field type (**Free-Form Text** holds 300 characters, **Text Area** holds 4,000, **Rich Text** holds 10,000, and **Long Text** holds 1 million) and also that only **Rich Text** fields include formatting using HTML, such as `` for bold and so on.

New in 2024, NetSuite has started to incorporate AI features into the system, starting with **Text Enhance** for many text-type custom fields. With an eye toward saving users' time when filling in text fields, Text Enhance helps people fill in text values for fields automatically using built-in generative AI. The system's **large language models (LLMs)** can assist with setting things like item descriptions, sales messages, job descriptions, and more. For instance, an HR person can get Text Enhance help with writing *Kudos* for employees, or with writing a case description for any commonly occurring shipment issues. This is built into every account now and available right away to all users.

Another common field type is **List**, and we define the list's values separately from any field that might use that list. NetSuite includes a few handy features you should know about to help us maintain those list values over time. So, for instance, you know what will happen when you need to remove one value from a list and add a new one. Use a custom list when you have just one value to present or create a custom record type when each of those values will have other related data that goes with them. For instance, if a client needs a custom **Handling Status** list, and the values will be **New**, **In Process**, and **Completed**, then use a list. But if they want to track a special-purpose code to go along with each status, use a custom record type to store the codes next to each status in the list.

There will be times when you have more than one way to capture some data, and NetSuite offers more than one good option. Here are a few examples and suggestions for how to resolve any uncertainty.

Checkbox versus list

Your client says they want to add a field called **Manager Approved** to a form, and they want it to be a list with **Yes** and **No** values. We generally recommend they use a checkbox in these cases instead, since **List** fields always have a blank default value, and it's never clear what *blank* means in a field like this. A checkbox, on the other hand, is either selected or it is not.

Single-select lists versus multiple-select lists

Once, a client of mine wanted to add a field to their item forms called **Allowed States**, because some of their items could only be sold in some states. We talked over the options and settled on multiple-select fields for this purpose. They had initially asked whether this should be a sublist of custom records so they could update the list whenever they needed. They thought they might start to sell these items in Canada, in addition to the US, for instance. We showed them how they could update the values in multiple-select lists, and how it takes up a lot less space on the screen than a sublist with a set of row data.

For visual reference, here is a typical sublist, with multiple fields on each row of data:

Contacts

Remove all

Clear All Lines

CONTACT*	JOB TITLE	EMAIL	MAIN PHONE	SUBSIDIARY*	ROLE
John Yannoerr				Parent Company	
Sally Tragerst				Parent Company	
Marlo Earpils				Parent Company	
<input type="text"/>				Parent Company	

✓ Add

✕ Cancel

+ Insert

🗑 Remove

Figure 9.6 – The default Contacts sublist on sales orders

Here is a multiple-select field with just one value per row:

The screenshot shows a NetSuite interface with a dark navigation bar at the top containing tabs: Items, Shipping, Billing, Accounting, Relationships, Communication, and Custom. The 'Custom' tab is active. Below the tabs, a form section titled 'FREE MOVIE SELECTION' contains a multiple-select dropdown menu. The menu is open, displaying a list of options: '- New -', 'Gone with the Wind' (which is highlighted with a blue background), 'Raising Kane', and 'To Have and Have Not'.

Figure 9.7 – An example of a custom multiple-select list field

You can see that the multiple-select is preferable if we just want to keep track of that list of selected values.

Image versus inline HTML

Image fields are OK – they make it simple to add an image to any record such as an employee – but they are not very flexible. I usually recommend using an inline HTML field instead, since they can include an HTML `` tag as well, and then you can resize the images to a standard size, position, and so on. It is not too hard to set up a default value in these fields, which keeps most users from having to worry about those technical details as well. For example, you might use something like this to define a link using an image in the file cabinet: ``.

For all of these field types, when you're creating new fields, always remember to give each a unique, memorable field ID, so you can use them in workflows, datasets, and so on (never let NetSuite assign a default ID, in other words), and also take the time to enter the **Help** text. This is useful for your new users later on, to remind them of what they're expected to do with each field.

As we discussed in the SuiteBuilder section earlier in this chapter, it is easy to add a lot of custom things to an account, but it is harder to manage it all and keep it easy for users to navigate these screens and access the fields/lists when they need to. This is where a solid change management plan really helps.

Change management for all your custom objects

If I could go back in time and counsel my clients on one thing they should do differently in their implementations, it would be to do a better job of managing (and limiting) the changes they make in their accounts.

NetSuite's amazing flexibility is great up to a point, but you know you have overdone things when users start to complain that they can never find the fields they need to set values in, or when nobody knows what a set of fields on a screen is used for, if they are used at all.

It is for these reasons and more that developing a solid, workable change management plan is such an important part of every NetSuite implementation. This process can be quite simple or much more formal and complicated, depending on what the business wants and needs. I have worked with clients who were small enough that we could just write up a single Word document, describing all the custom configuration and automation we created for them, and then they just referred to that whenever they had a question about what was happening in their account, or whether they could change something safely.

But many companies are large enough, and have many more users in many separate functions, trying to keep a document like that up to date and accurate would never work. These businesses need something more robust, and they need to be able to keep it up to date as things change (which they always do). With a client like this, we usually recommend creating a custom record type in the production account for each of the following topics mentioned.

Requirements or use cases

It is important to start all the enhancements users want in the system with a description of the change and a statement of the business need explaining why it needs to be made. The business owners from the related department will typically have things working pretty much as they need when the company goes live on the system, but there are almost always things they want to tweak. The change requests and new features they are looking for should be documented as requirements or use cases. For instance, if the **customer service representatives (CSRs)** wish they had a field for tracking the customer's secondary email address on **Cases**, that is an enhancement request and so has requirements or a use case.

Issues or defects

When NetSuite features are understood to work a certain way but do not, we have an issue to be tracked until all parties involved agree it is resolved. We use a record type with a relevant list of fields on it to keep track of these issues and to be able to report at any time on their current status, who owns the next action, and when they are expected to be resolved.

System change

When we know we want to change how a field is defined in the account, or we are going to remove a field from a set of forms, it is a good idea to record that change in its own separate record type. We can also then define a process around managing these changes so that, when the change needs to happen in the live production account, for instance, we will be able to record every change we make (as well as have a place to review later) if issues should arise. This can be really helpful too when it comes to moving custom things such as fields and lists from one NetSuite account to another. We call that process **deployment** or **migration**, and it's very important to know where each version of each custom object is. I'll cover this in detail in *Chapter 18*.

It does not make sense to enforce the use of all these records early on in an implementation project since there are so many changes being made every day. However, as you get remarkably close to going live, using records like these becomes critical – especially in larger, more complex jobs. Work with your client to get a sense of how much of a formal process they want/can support and define these records to meet their needs. I will also tell a client about a process I think is more formal than they really need, just to have the idea out there and to know we can go in that direction should things get out of hand with the simpler, more straightforward approach we start with.

Keeping track of changes as they occur can be difficult, and it gets more difficult as you add more members to either your own team or the client's list of implementation people. But the benefits of maintaining this information over time will far outweigh that extra work. The alternative, where it is difficult to know why any one feature or setting in the account is configured in a certain way, or whether it is even used at all, can lead to a lot of problems. Know this upfront and work to avoid that kind of drama by routinely using the ideas outlined in this section.

Summary

In this chapter, I hope I've taught you more about customizing your clients' NetSuite accounts using forms, records, fields, and more. Forms hold the set of fields, lists, subtabs, and sublists together in the arrangement that works best for the users. Custom record types are added to the list of records NetSuite ships with, allowing us to store new types of data specific to the business using each account. Fields give us a place to store additional data on the native and custom records in the system, in a wide range of formats. Your job is to add these things to each client's account in just the right measure to allow them to run their business effectively and efficiently.

Always keep in mind that your goal is to make NetSuite as simple to use for the users as you possibly can, while still meeting all their data and business requirements. We use the form design tools, centers, and dashboard features to streamline the day-to-day operations of all the user groups at the company, which we will learn about in the next chapter. This makes for happy users, which makes for satisfied managers when the project is done.

Self-assessment

Take a moment now to think back over this chapter's contents and test your understanding of the topics covered. Think about how you would resolve these challenges in your work:

1. Your client lets you know that they need to be able to define custom kits, each of which is its own collection of items, sometimes sold together and sometimes sold separately, and the contents of these kits vary all the time. Assuming the native kit-type items will not meet their needs, which SuiteBuilder features can you use to help meet this need?

-
2. A software company wants to be able to record sales orders for both contract-based sales and non-contract sales items. The fields on the two types of orders will be quite different and they do not want users completing non-contract sales to have to deal with a lot of contract fields. How can we use forms in NetSuite to help with this?
 3. A client wants to require some customers to provide a **personal identity number (PIN)** unique to the products on their orders. It is a 4-digit number they assign to the customer when they are registered, and they need to provide it to a customer service representative when they place an order. The PIN must be stored on the order, but it cannot be visible to any user. However, it does need to be accessible from SuiteScript for validation. How can we meet these requirements in a NetSuite account?

Centers and Dashboards

NetSuite's **centers** feature allows us to group features according to who will use them. Each NetSuite account includes the Classic Center for general access, the Sales Center for salespeople, the Support Center for knowledge base and case tracking, and more. The Employee Center is especially popular, since it allows people who would not otherwise need to interact with NetSuite a way to enter their time, book time off, and more. Altogether, these centers are a popular extension to NetSuite's main interface, and so learning how to use one or more of them is key to most implementations.

Dashboards are the screens most users see when they first log in, and they can be customized to fit the needs of each group of users a client has. In this chapter, you will learn how to set up these centers and dashboards for your user groups.

Here, you will learn about the following topics:

- Setting up native centers
- Using custom centers for even greater control over the **user interface (UI)**
- Use case – setting up and using the Customer Center
- Setting up dashboards for groups by role

As you work through these topics with your clients, you will want to engage SMEs from each department to make sure they're satisfied that their users' centers and dashboards are set up as efficiently as possible.

By the end of this chapter, you should be able to configure your clients' native and custom centers and dashboards, allowing users to get straight to their jobs every time they log in. Read the Help page on the subject of *Centers* (https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_164208316196.html) to become familiar with these features, if you have not had the chance to work with them before.

Setting up native centers

When we look at a brand-new NetSuite account, we typically sign in as the so-called *Primary Administrator* user. That user is assigned to the Classic Center by default, and so we see the following menus across the top of the UI's main screen:

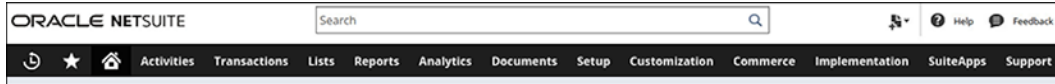


Figure 10.1 – The default menu bar in a new NetSuite account

Your screen may not match this exactly, since NetSuite is very flexible and configurable, and because each account ships with various options enabled at the time it was created. Generally, though, this is what the Classic Center looks like. With this center, users such as administrators quickly learn to look at the **Transactions** menu for things such as orders, and the **Customization** menu for the **SuiteBuilder** features. But this can be a lot to take in for many users, and it includes features that most people won't need or won't have permission to access. NetSuite takes care of hiding things the user doesn't have permission to view at all, but we can still do our own configurations to clean things up further.

The system comes pre-built with a long list of additional centers, each of which is fine-tuned for its intended groups of users. You can see a full list in the Help section by searching for the *Centers* page (at the URL in the previous section). That list includes entries for accounting specialists, executives, sales personnel, support people, and more. As a consultant, you should learn to use those centers as a starting point so that you don't have to reinvent the wheel for each new client you work with.

Each role you set up for your clients will be associated with one of these centers. When you customize one of the system's native roles, such as **Sales Administrator**, it will have a center assigned already, and you won't be able to change that (**Sales Center**, in this case). But when you create a new role from scratch, you can choose which center the role's users will be associated with. For instance, some users will appreciate having access to the features included in the Sales Center or the Support Center, and some will want to use the more common Classic Center. We use this selection on each role as a quick way to streamline setting up the users' experience in NetSuite, by limiting menus and other UI choices to just those features they will use on a daily basis.

Beyond the standard centers, NetSuite also offers access to a few other centers, which are meant for people coming to the account who would not otherwise be full NetSuite users. These are extra or special centers and are mentioned in the sections ahead.

Customer Center

A default example of the **Customer Center** home page may look like the following:

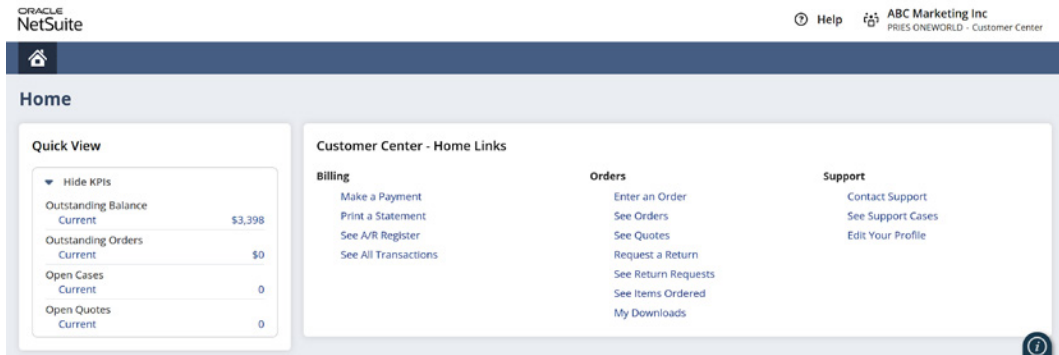


Figure 10.2 – Example of the Customer Center default home page

This is how your client can let their customers sign in to the system. NetSuite sells these as separate licenses, and they are automatically restricted in what they can access. Instead of the Classic Center UI, they see only the features the system allows them to, including sales orders (creating and viewing) and invoices.

Employee Center

This is another separate UI that NetSuite provides with a limited set of enabled features, and this one is great for use by employees who would not otherwise need to sign in to NetSuite (and so won't take up a full user license):

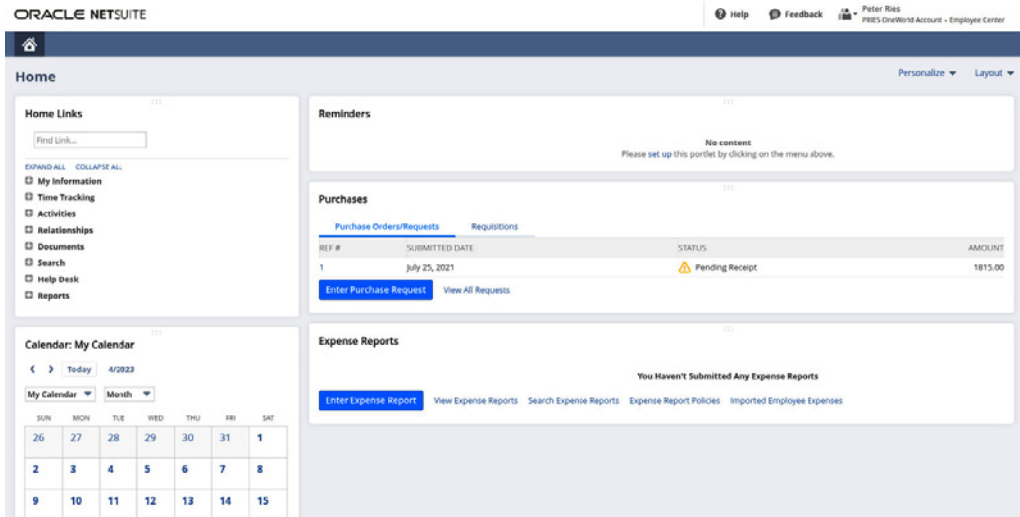


Figure 10.3 – Example of the Employee Center dashboard

These employees can do things such as submit time-off requests, file expense reports, and enter their timesheets. They can also submit purchase requests and view certain types of reminders and reports related to these same features.

Partner Center

An example of a **Partner Center** dashboard may look like the following:

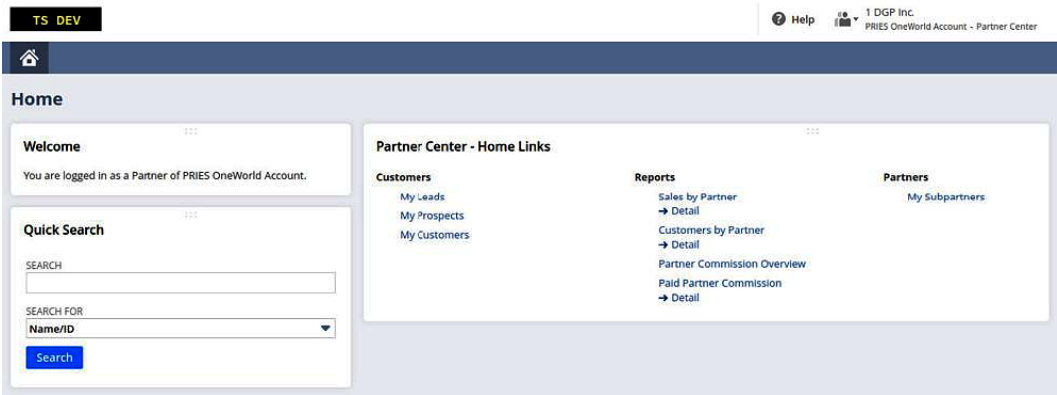


Figure 10.4 – Example of the default Partner Center dashboard

This center is for your client's partners, allowing them to log in to access features if they need to, such as sales and promotions data. This is not used very often in my experience, but some companies do need it, and if so, they might want to consider the even more useful **Advanced Partner Center**, too.

Vendor Center

The **Vendor Center** dashboard may look like the following:

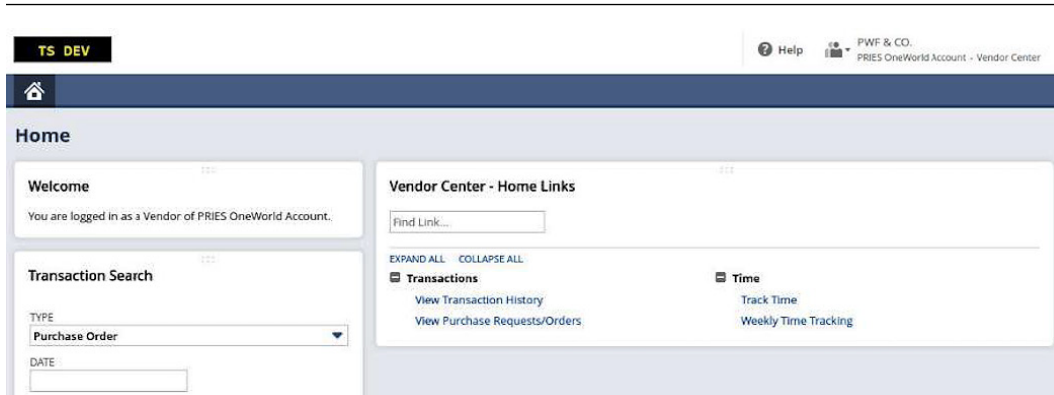




Figure 10.5 – Example of the default Vendor Center dashboard

 **Quick tip:** Need to see a high-resolution version of this image? Open this book in the next-gen Packt Reader or view it in the PDF/ePub copy.

 **The next-gen Packt Reader and a free PDF/ePub copy** of this book are included with your purchase. Scan the QR code OR go to packtpub.com/unlock, then use the search bar to find this book by name. Double-check the edition shown to make sure you get the right one.



Vendor Center allows your clients' suppliers to log in and access things, such as the purchases your clients place with them and the history of payments applied to those.

All the centers mentioned come with a matching standard role in NetSuite, and you should create custom versions of those before you assign people to them. You can customize some features for each role, but keep in mind that NetSuite generally sells these roles much cheaper than a full user license, so they will always be very limited in what they can do once you are logged in.

Next, we'll talk about the special reasons you might have to create a custom center in an account.

Using custom centers for even greater control over the UI

For some clients, you might find that you need to create a special solution, maybe with lots of **SuiteBuilder** custom configurations such as records, fields, and lists included in it, and maybe you'll add scripts or workflows to that mix. When this happens, we typically want to set those screens aside in a special way, making it clear to users how they can access the special functionality, and that's when we create a custom center.

For instance, let's use this screenshot of the **Documents** menu as a reference:

Documents	Setup	Customization
Documents Overview		
Files >		
Templates >	Email Templates >	
Mail Merge >	Fax Templates >	
	Letter Templates >	
	PDF Templates >	
	Template Files	

Figure 10.6 – The default Documents menu with templates from NetSuite

The built-in **Documents Center** is a container for everything under that menu. The tab for this center is at the top, labeled **Documents** in this example. The center's categories are **Files**, **Templates**, and **Mail Merge**. The center's links are the topics under each category, such as **Email Templates** under the **Templates** category.

You can create entirely new custom centers when you need to, and those will include lots of useful features, such as a role-based audience list and permissions. Check out the Help pages on *Creating and Editing Custom Centers* for more information, and use this to make sure your clients' users see just the features they really need access to, including your customizations.

Once you've mastered the natively available and custom centers, you can move on to using the extra centers I described in the previous section. Let's see what setting up one of those – the Customer Center – is all about.

Use case – setting up and using the Customer Center

As a deeper dive, let's explore what is involved in setting up the Customer Center. NetSuite sells licenses for this UI to businesses that want to let their customers log in to a limited version of the system and access just the things they create within that context. For customers, that means sales orders, invoices, and payments they've made.

Setting this up is simple once you've purchased the licenses from NetSuite. First, we need to create a custom copy of the standard **Customer Center** role. That's done in the **Setup | Users/Roles | Manage Roles** screen. When customizing the role, it's generally a good idea to start simple; don't make many changes at first. Then, set up a customer with access to use that role/access the center, and test that a bit with your client so that they can see what the experience is like in its default state. They will almost always find things they want to tweak about the customer experience, and then you can change the role to suit their needs.

You can see the default **Customer Center** home page in NetSuite in the following screenshot:

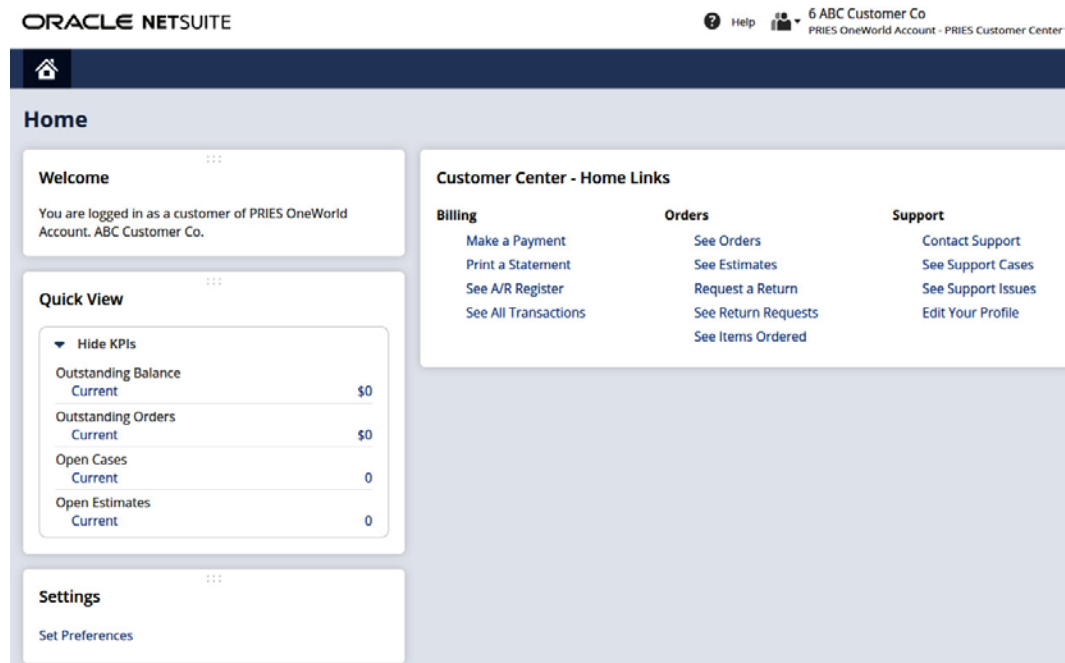


Figure 10.7 – The default Customer Center home page in NetSuite

All of the features available to Customer Center users are right there on the **Customer Center - Home Links** screen. You can see that the customer's name and the name of the account are shown in the upper-right corner, and the UI is generally very similar to the one shown to every full NetSuite user. Clients very commonly ask how much of that can be customized, and the answer is *not much*. Because the license is sold at a lower price than a full user, NetSuite doesn't allow a ton of customization.

If your client decides they really like this idea (allowing customers into the system to access their data directly), but they want to customize the look and feel of the UI more, another more powerful option is to purchase the **My Account** portion of **SuiteCommerce Advanced**. We've had clients who purchased that add-on and then didn't use the e-commerce store portion at all; they just wanted to be able to customize the UI customers see when they visit their **My Account** screen. Talk to your NetSuite salespeople about this if you find a client with similar requirements.

Next up, once we start using the centers that come with NetSuite, we typically also need to customize the home page for each user group, and that's where dashboards come in.

Special highlight – SuitePeople and employee management

Just a few years ago, NetSuite introduced the human resources add-on, **SuitePeople**, and it's been growing in function and features ever since. The solution extends a NetSuite account and includes three main modules, covering what most organizations need to manage their employees' time, expenses, and payroll (and much more): human resources, workforce management, and payroll (USA only). With SuitePeople, companies keep repetitive and manual processes to a minimum by making NetSuite the central location for all employee-related data. The solution adds a lot of new data to each account and includes tons of analytics and reporting as well. NetSuite sells this as an optional feature that any client can add to their account, and there is usually a separate implementation process, which typically starts after the company is live on the product for CRM and ERP. Each of the modules I mentioned previously adds a slew of new features, so clients with many employees should always check out SuitePeople.

Setting up dashboards for groups by role

When you log in to the system, by default, you're taken to the home page. In a brand-new account, that's the system's default page for your role. That includes everything in the Classic Center (as discussed in the previous sections) and also all of the sections on that screen with their various content.

Here's an example of a dashboard in NetSuite:

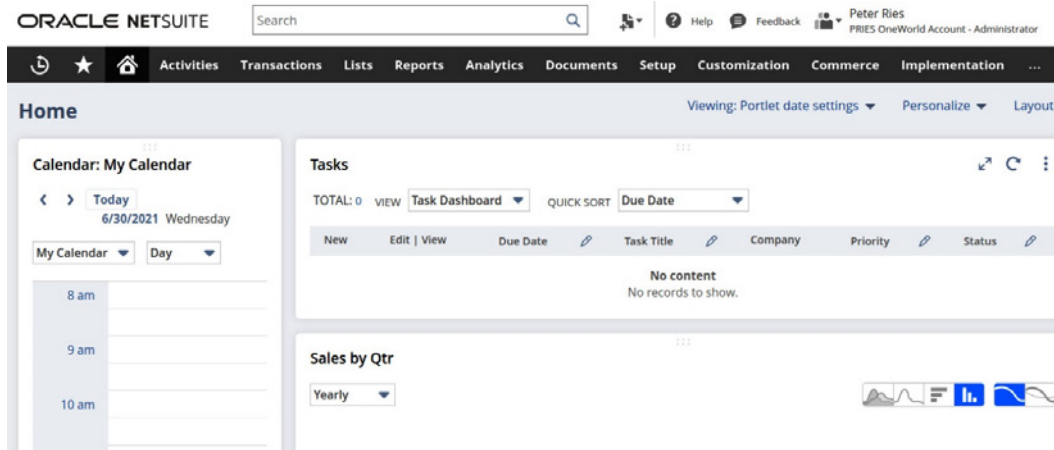


Figure 10.8 – A personalized dashboard on the home page

Those rectangles of content are known as **portlets**, and any user can click the **Personalize** link at the top right of the screen to add or remove them as they like, or to move them around. Portlets can include pretty much any data you have in the account in a variety of formats, including charts, tables, lists, and more.

When we're helping a client decide what their users' dashboards should include, we need to focus on just their daily activities – the NetSuite features they will access regularly. That thinking should drive you to create dashboards for each role: accountants, salespeople, support reps – everyone. Remove the features they won't use and add shortcuts and reminders for the things they will use. This can create “*Aha!*” moments, when an SME or new user goes from looking at the default screens with lots of links to things they don't need, to seeing a well-crafted dashboard with everything they do need, right there at hand.

Consultants and/or client administrators can accomplish this by setting up and publishing dashboards for any or all roles as needed. The process for publishing (or sharing) a dashboard is pretty simple. You log in with an appropriate role, then set up your own dashboard any way you like. There is a link on the same screen, allowing you to publish the dashboard to selected groups of users via **roles**. While you're there, you can also restrict their access to change or personalize the dashboard, but we usually find that users are happier when they can adjust things based on their own needs, once we get them started with the starter version of the dashboard.

Check out the ample Help pages on setting up and *Publishing Dashboards* for more of the *how-to* information you need, and check out the script type known as *portlet scripts*, since, with these, you can pull in data from external sources, mix that with data from the account, and much more.

Here are some additional ideas to use dashboards productively with your clients:

- Create searches for any lists of data people need to see on their home page and include those in a search portlet. For example, if the customer support reps need to see a list of their assigned issues, make that list easily available to them on their home page, and give them filters so that they can get the list down to just what they want to focus on, from moment to moment.
- Don't forget your admin users – create dashboards with things such as NetSuite user login history, scripts, and web service error logs so that they can spot any trouble in the account quickly from their home pages. Include elements from **SuiteCommerce** too, when that's in play.
- Learn how to use the charts and graphs the system offers, but always use them to communicate really important information. Focus on the actions users can take when they look at those charts. Numbers and colors are pretty, but you're only helping the users when you deliver things that help them do their jobs.
- Teach every administrator and as many individual users as you can how to customize their own dashboards! They know what they need best, and each user will bring their creativity and intelligence to the task in ways you won't have even thought of. They'll also help others solve their problems once they know how.

Think of the dashboard as a canvas for information and use its many features to paint the most informative and actionable picture you can, for each user group your client has logged in to every day. Remember that the home page is each user's first impression of the system, and they tend to look at whatever you present them with as *this is what NetSuite is and does*, so take the time to do this well and you'll successfully convince them to dig deeper into using the system.

Summary

During implementation projects, we're typically on a very short schedule, and time is always tight. It might be tempting to skip a lot of customization and configuration around the things covered in this chapter – centers and dashboards. You could, for instance, just make sure the roles are set up with the right default center and then use the default dashboards that come with those. You could tell the client they need to figure out how to customize those things, or (worse!) tell each user they need to do this for themselves. But that, in my opinion, is the wrong approach.

As administrators of the system, we have a responsibility to use the advanced features that NetSuite offers to really turn the system into a much easier-to-use and efficient tool, allowing users to get more work done in less time. These features can do that if they're set up well by experienced guides such as you. This fits into the default Waterfall method we usually follow for new implementations as well, since we want to get this right at this stage, so we can rely on using this proper configuration for everything else we still need to do.

In the next chapter, we'll move on to talk about how we set up something more tangible and immediately useful to users – items, including all their variations.

Self-assessment

Take a moment now to think back over this chapter's contents and test your own understanding of the topics covered here. Think about how you would apply these concepts in your day-to-day work. Consider the following questions:

1. You know you're to set up your client's sales roles with access to the Sales Center since it is designed with them in mind, but they ask you for more customization to the screens, fields, and so on. What other options does NetSuite offer to enhance the salesperson's experience of the system?
2. Your client has a group of employees who install their hardware and software in the field. They're located in regional offices all over the US. They don't need to sign in to the system for operations on a daily basis, but they do need to be able to share a calendar so that everyone can see who's going to be on vacation each week. How can you help them with this?
3. As part of an implementation, you create a custom solution for a client, called ABC Company Warranties and Returns, which consists of a set of searches, custom records, scripts, and workflows. How can you help users easily find and use this new functionality in their account, all in one place?
4. Your client wants you to set up the Customer Center for them, but they only want their customers to be able to see their invoices and no other transactions. How can you set that up for them?
5. A company's small group of sales managers needs to see a list of orders they've submitted that require their approval. To make this process as simple as possible, they've asked whether you can add the list to their NetSuite home pages and allow them to approve or reject the orders right there. Is this an option with dashboards and portlets?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search for this book by name.

Note: Keep your purchase invoice ready before you start.



Items and Related Lists

Items are a key part of every NetSuite implementation since they represent more than just what the business buys and sells. They are also used to define the fees, discounts, and other accounting-related things the company needs to keep track of. Every consultant needs to learn how to correctly set up items, as well as related lists in NetSuite, to avoid problems or confusion later.

In this chapter, we will cover the following topics:

- Enabling item types, inventory management options, and so on
- Defining the item forms and fields
- Defining purchase prices and sales pricing for items
- Setting up matrix item types and item options
- Setting up other item types as needed
- Special feature: **Supply Chain Control Tower**

You will be working primarily with the people who have been put in charge of owning the list of items from your client's SMEs for the work in this chapter. Those people might be from the IT department, the product team, or elsewhere within the company. They each need to understand how to add, edit, import, and remove items by the time they get through this part of the implementation.

Enabling item types, inventory management options, and so on

Starting from your client's requirements (which you should have already gathered, as described in *Chapter 7*), you can go into their account and start to enable the item and inventory features they need.

You do this via the **Enable Features** screen, which can be found via **Setup | Company | Enable Features**. You should have previously visited this screen with the client to make basic system-wide choices and set up the accounting features. Now, you can use the **General** tab to make choices that will affect the business use of items, such as whether they need the **Multiple Units of Measure** and **Multiple Currencies** features. Then, visit the **Items & Inventory** tab to make further configuration choices.

Items versus inventory

In NetSuite, there's a big difference between items and inventory, so I need to be sure we're clear on those terms. Items are the collection of settings that define each thing a company makes, buys, is charged, or sells/charges for. Examples of items are things such as "Steinway Grand Piano," "10x6 plywood panel," "Late fee," "Professional Services charge," and "Subscription fee per megabyte." Inventory is how NetSuite keeps track of how many of each item you have in stock at each location. There are a variety of item records for keeping track of your items, and a variety of transactions for tracking your inventory as it comes in and goes out from stores or warehouses.

There are approximately 40 choices on the **Enable Features** screen for **Items & Inventory**, and we cannot cover them all, but it is important to carefully consider each of your choices as you work your way down this list. Which of these you enable will vary with your client's industry, and also with how they will use items within the system. Make sure you are familiar with each option. You need to know, for instance, which native features will cover your client's requirements.

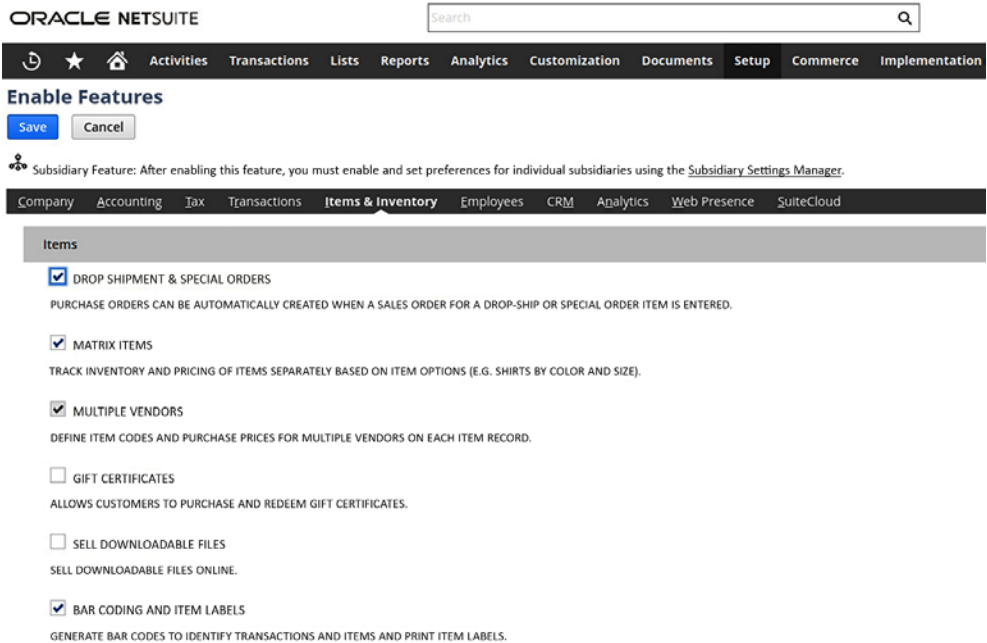


Figure 11.1 – The Enable Features screen for Items & Inventory choices

For instance, with a feature such as gift certificates, the implementation of NetSuite is generally limited and only works in one way. The system thinks of gift certificates as single-use items you can sell to someone today and then the recipient will redeem the gift certificate later on. You must know how closely your client’s requirements match a feature of the system before you decide to enable it here.

This is because if the native feature does not help your client, then you are better off leaving the feature disabled (as it is by default) and replacing it with some other solution (such as one from a SuiteApp/ partner or customization you create for the client specifically). The users should not have to guess which **Gift Certificate** screen they should go to – the native one or the custom one – so do not enable the native option if they will not use it at all.

Some of these options require you to enable other, related options or make irreversible changes to the account once enabled. For instance, **Inventory Status** requires the use of the **Advanced Bin/ Numbered Inventory Management** option. Enabling the **Advanced Bill of Materials** feature makes changes to various item screens that are difficult to undo later. If you have access to one, it is always a good idea to try changes to features like these in a sandbox account first, either yours or the client’s, to make sure you understand the full effects each change will have on the account.

Once you have enabled the features your client needs, the next step is to start defining item forms.

Defining the item forms and fields

We will start the defining process by reviewing the forms for each of your client’s critical item types – **Inventory**, **Non-Inventory**, **Service**, and **Discount** – and making sure they are set up to work as per the client’s requirements. (This can be done by going to **Customization | Forms | Entry Forms**.)

Here is what the default form screen looks like for **Inventory Part** items:

The screenshot displays the 'Custom Entry Form' interface for 'Inventory Part' items. At the top, there are buttons for 'Save', 'Cancel', and 'Save & Move Elements'. Below these are input fields for 'NAME' (set to 'Custom Inventory Part Form') and 'ID'. The 'TYPE' is set to 'Item' and 'SUBTYPE' is 'Inventory Part'. On the right, there are several checkboxes: 'ENABLE FIELD EDITING ON LISTS' (checked), 'STORE FORM WITH RECORD' (unchecked), 'FORM IS PREFERRED' (checked), 'USE FOR POP-UPS' (unchecked), 'POPUP ONLY' (unchecked), 'USE FOR MANAGER' (unchecked), and 'WEB SITE CONTENT MANAGER ONLY' (unchecked).

Below the form configuration, there is a navigation bar with tabs: 'Subtabs', 'Field Groups', 'Fields', 'Actions', 'Sublists', 'QuickView', 'Custom Code', and 'Roles'. The 'Fields' tab is selected, showing a list of fields with columns for 'DESCRIPTION', 'SHOW', 'MANDATORY', 'DISPLAY TYPE', 'LABEL', and 'FIELD GROUP'.

DESCRIPTION	SHOW	MANDATORY	DISPLAY TYPE	LABEL	FIELD GROUP
Custom Form	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Normal	Custom Form	Primary Information
Internal ID	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Normal	Internal ID	Primary Information
Item Name/Number	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	Normal	Item Name/Number	Primary Information
UPC Code	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Normal	UPC Code	Primary Information
Display Name	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Normal	Display Name/Code	Primary Information

Figure 11.2 – The default inventory item form with the Fields list

As I have mentioned in prior chapters, take the time to remove anything the users do not need to see while you are doing this. Create a copy of each of the standard forms with a name, including the client's initials, and save each item type form. Some clients will need multiple forms for one type, such as for **Inventory** or **Assembly Items**. We always try to stick to just one form per item type, but special circumstances drive us to solutions like this sometimes.

Once the forms have been set up for every item type, you can start talking to the client about setting up items. In most cases, we first show users how to set up an item manually a few times, by clicking around in the UI, since this allows them to become familiar with the lists and fields, but then we perform CSV imports for the majority of the items list. This ties into the forms' designs too, since we want the list of fields they include in the CSV import to be a close match for the fields and lists that are available on the form.

We typically provide a CSV template file for their use, in the form of an Excel file, and that should include a set of header rows with explanatory text. This helps them understand the values that will be allowed for each column and gets them up to speed on the native CSV import format NetSuite will expect from them. We like to make these imports the client's job to perform for themselves because this gives them hands-on practice with interacting with the system, and it also makes them responsible for the success of the imports. That is critical because, otherwise, some clients will stand back and hope that their implementation team will complete tasks like this for them. However, they need to see that they can learn to do things like this themselves, with coaching and demonstrations from you, of course.

Depending on the overall length and complexity of your project, a first version of the full items list should have been imported into the account about 20-30% of the way through the project. This is because you cannot import the full items list until their requirements are fully understood, yet you need to complete this step before you can move on to other things such as setting up customers and transactions in the account later.

It is not uncommon for us to realize we have a problem with the way we imported the first set of items, by the way, and in those cases, we erase them (where NetSuite allows) and re-import them, with the data corrected in some way. Just remember that NetSuite will not allow you to completely delete an item with any transactions posted using it, so you might have to remove those first, adding to the time and effort it takes to fix mistakes at this stage. A good amount of planning is needed to avoid these costly mistakes, but many of the system's finer points need to be learned firsthand.

Once you have at least a starter list of items in the account, you can add additional details to them, which is common when we must address costs and prices for each item. Let's look at that next.

Defining purchase prices and sales pricing for items

You can establish the prices your client will pay their suppliers for each item and the prices they want to charge at the same time as you import items. I have found, however, that many clients have more complex requirements around these two topics, so we typically do this as a follow-up exercise. We define the basics about the items first and then export those item lists from the account as CSV files. Then, we use those files as the starting points for the purchase price and sales pricing updates we make next.

Purchase prices can be defined per item in several different ways, and how each client wants to do this will vary depending on the item's type, their business needs, and more. The native **Purchase Price** field is available on every inventory and other item type screens if the client is OK with having just one price per item within the account. Populating this field is optional, though; if we do not set it per item, then NetSuite will set a default price on purchase orders based on the most recent purchase order for the same item. We can take this a bit further by setting purchase prices per vendor on the **Purchasing/Inventory** sublist. This is useful when the business buys the item from more than one vendor.

NetSuite will automatically update **Last Purchase Price** using the costs entered into inbound transactions such as **Item Receipts** and **Inventory Transfers**. If your client has landed costs to factor in for their items, check out the Help page called *Entering Item and Cost Detail* for more information on all the features they can use to help with this.

With the Procure-to-Pay (PTP) prices set, let us talk about the Order-to-Cash (OTC) prices next. Sales pricing is more complicated. First, we can set a **base price**, MSRP, or whatever the client chooses to name this default *price level*. Then a client can define as many additional price levels as they need, which can either set a price as a percentage of that base price or be manually set to any arbitrary amount. These amounts on the item screen control the default for each item, but the actual price that we will see the system use on the various sales transactions is based on a more complex set of rules. We will go over that in detail when we get to *Chapter 15*.

But for now, just know that sales pricing options for items are fairly flexible, and we should always try to convince each client to stick with the set of native pricing features, if at all possible (to avoid having to add a customization, if possible). This includes setting the **Item Group** field on the item and using **Quantity Pricing Tiers** per item, allowing the company to price higher quantities at a lower price, for instance. Item administrators can manage the prices on items via the NetSuite UI (one item at a time), or for bulk item changes, they can do this via **CSV Imports** or **Mass Updates**.

In this highly competitive world, though, we know that many companies are successful only when they carefully set their prices versus their competitors, so the native features will not always get us where we need to be. Pricing customizations are fairly common, and there are several SuiteApps and customization bundles available from NetSuite and its partners to help streamline this in the most common ways. For instance, check out *Effective Date Pricing* on the system's Help pages, or look for more solutions from your NetSuite contacts or the SuiteApp Marketplace (the **SuiteApps** menu in the NetSuite UI).

Next, let us talk about one of the more complicated item setups that can trip up the progress of a project, that is, if we are not aware of all the potential problems that they can cause if they are not handled correctly – matrix items.

Setting up matrix item types and item options

When a client needs to track multiple variations of an item, we have a couple of options we can recommend, depending on the situation. First, you might start by suggesting that they use **Item Options**. This is simply a list of things we can assign to any one item. We might, for instance, have an item called *100% Cotton Hoodie*, and we know we can sell it in red, blue, or orange styles. So long as we just have a simple set of variations like this, item options can work fairly well. We just set the list of options on the item’s screen and then we can select from that list when creating a sales order later on.

This can be too limiting, though, for many items, and in those cases, we need to go with the more complex matrix-type items. With matrix items, the system allows us to define a parent item, the hoodie in the preceding example, and then a set of child matrix items, each representing one variation of that parent. For instance, if we need to set up and sell separate items for color and size variations, we will create children for the hoodie called Red – XL, Red – L, and Red – M. We can create these items ourselves via a CSV import, though NetSuite offers **Matrix Item Assistant**, which walks us through the process of defining the parent and the variations and then creates the child matrix items for us automatically.

This is what that assistant looks like:

Matrix Item Assistant

STEPS

1 Set Up Item Basics

2 Create Item Property Lists

3 Choose Property Combinations

4 Choose and Create Items

Set Up Item Basics

Provide basic matrix item information below. Advanced configuration options will be available in subsequent steps.

ITEM NAME/NUMBER *

100% Cotton Hoodie

DISPLAY NAME/CODE

100CHOOD

VENDOR NAME/CODE

SUBITEM OF

SUBSIDIARIES

Parent Company

Parent Company : PRIES Sporting Goods

☐ INCLUDE CHILDREN

SALES DESCRIPTION

PURCHASE PRICE

COSTING METHOD

Average

SCHEDULE *

Figure 11.3 – The first screen of Matrix Item Assistant

Assuming you have decided you do need to use matrix items, this is a good screen to walk your item experts through, since it should help the users understand all the details that go into defining an item of this type. They will see what makes matrix item types different from non-matrix types. They can then manage CSV imports better, in my experience (and defining matrix and non-matrix items is how we usually handle the CSV imports for a full item list).

While we are on this topic, we should take the time to explain how using matrix-type items is more complicated for users who do things such as create orders, track item availability, and so on. They're fine to use, when needed, but then I stress to users how they add complexity in data entry and reporting. It is important to show the SMEs how they will select a matrix item on a sales order, for instance, and how matrix items affect their various inventory and financial reports. This is true whether you decide to use matrix items right from the start or have to switch to using them later in a project.

The system supports many other types of items, of course, so let us review setting up and using a few of those that are commonly implemented.

Setting up other item types as needed

Depending on the features you have enabled for your client's account, they may need your help getting a few other item types set up. The following sections cover the most common of them, but just know that these tend to be simpler to explain and get the client up and running with.

Description

Create a **DESCRIPTION** item when you want to include a text message on transaction lines but have no price or other details associated with it. For instance, we have had clients who wanted the users to be able to tack on a line to their orders, saying something like `You received free shipping for the item above!` in some cases.

Discount

When we define a discount-type item, we do not tell NetSuite how the discount will work (for example, 10% off) and which items it can be used with – we do all that on a separate screen. Discount items are a part of promotion tracking in NetSuite. Some companies will only need the simple promotions the system offers natively, but many need the more robust **SuitePromotions** SuiteApp when they require control over the discounts being applied.

Here is the screen we will see when setting up a new discount-type item:

Discount 🔍

Save Cancel

Primary Information

ITEM NAME/NUMBER ★
 RATE ★
 UPC CODE
 DISPLAY NAME/CODE

VENDOR NAME/CODE
 SUBITEM OF
 PRODUCT

DESCRIPTION

Classification

SUBSIDIARY
 Parent Company
 Parent Company : PRIES Sporting Goods

DEPARTMENT
 CLASS

LOCATION

☐ INCLUDE CHILDREN

Figure 11.4 – The new Discount item screen is much simpler than for the other types

There are not a lot of fields to fill in here. We usually include some description of the discount in the **ITEM NAME/NUMBER** field though, such as 10% off or Buy one get one free.

Assembly items versus item groups versus kits

Assembly items, **item groups**, and **kits** are all based on a single item that has a list of components. They differ in how we track the inventory and the pricing for the item that is sold to customers.

Assemblies have to be *built* in NetSuite via a work order (and more, sometimes). Item groups are simply shortcuts that allow you to enter a set of items into an order (for instance) rather than making you select all of the component items. They are not stocked or fulfilled as a group, though. Kits are like groups, except that they can be sold as a single item (but we can choose to have NetSuite display the full component list on sales orders if the client wants to see that). Check out the Help page titled *Groups, Assemblies, and Kit/Packages* for more on this; it is very informative, and this is an important topic to know well as you consult with your clients.

Other charges

The name of this item type seems very generic, but these items have a specific use. It is typically software and services businesses that need to define these charges as part of their contract or subscription sales. They are simple to define in the system, and they are used in conjunction with SuiteBilling or another solution's sales process. For instance, if a company sells its customers a subscription to a service, and the customers can incur additional charges when they require additional services, that client will define an **other charge**-type item for this purpose.

Subtotal

Some clients want to control the list of items on their invoices very carefully, and in cases where they need to show a subtotal for any set of items, we use this item type. Setting these up is very simple, and NetSuite allows users to insert one into an invoice as needed. When we do that, NetSuite calculates the item prices and discounts above the **SUBTOTAL** line and sums them up. The remaining lines on the transaction will have their own subtotal, and then the total, if one is needed. I have found this to be occasionally needed for my *services or software industry clients* who send invoices with lots of lines for many items and then need to group them for one reason or another.

Inventory management

As mentioned earlier, tracking what you have in stock for each item at each store or location is what we call inventory management in NetSuite. That is done via a set of transactions, such as **Item Receipts** and **Inbound Shipments** (for tracking inventory coming into your locations) and **Item Fulfillments** (for tracking inventory leaving your locations). We'll cover those transactions more in *Chapters 14, 15, and 16*. But for now, keep this in mind.

Lastly, for this chapter, let us check out a relatively new feature offered in NetSuite for companies that manage supply chains for their items.

Special feature: Supply Chain Control Tower

NetSuite recently added a nice new feature for wholesale distributors, manufacturers, or any business that purchases or makes its inventory items and needs to carefully manage the supply and demand over time. The **Supply Chain Control Tower (SCCT)** is a combination of new features, including inventory snapshots for your inventory and assembly items, vendor performance reports, a predicted risks portlet, and a handy dashboard to gather all these details in one place. If your client thinks they need help managing the supply and demand for some of your most important items (or all of them!), they will want to check out these features.

To enable the features, visit **Setup | Company | Enable Features**. Enable the SCCT feature on the **Items/Inventory** tab and the **Predicted Risks** portlet on the **Analytics** tab. For each location where you want to use these features, enable the **Supply Planning** and **Supply Chain Control Tower** features. Then, allow at least a couple of days to pass, for new purchase orders, sales orders, work orders, transfers, and so on to be created in the system. NetSuite needs that time to pass so it can generate the expected data based on the business’ supply and demand over time. You can control how NetSuite controls its SCCT predictions and snapshot values by visiting **Setup | Accounting | Control Tower Preferences**.

A **Supply Chain Snapshot** is a recording in time of an item’s current inventory and its incoming supply (purchase orders, transfer orders, and work orders) and the outgoing demand (sales orders, transfer orders, and so on). You can either manually create a snapshot whenever you need – one item at a time – or you can set up scheduled automation to generate and refresh snapshots whenever you like. This is what a schedule for snapshots looks like:

ORACLE NETSUITE

Search

Activities SuiteTalk UI Payments Transactions Lists Reports Analytics Documents Setup Customization

Schedule Supply Chain Snapshots

Save Cancel

Primary Information

NAME *
SCCT Snapshot Schedule 1

DESCRIPTION *
Daily automation to generate snapshots for certain high-demand items.

☐ INACTIVE

ITEM SAVED SEARCH *
PRIES Items (Private)

Schedule

RECURRENT FREQUENCY *
Daily

REPEAT EVERY *
1

NEXT DATE *
1/2/2023

NEXT TIME *
6:00 pm 6:00 pm

Figure 11.5 – A schedule for supply chain snapshots

You can see that the schedule will be executed daily, based on a saved search for specific items. How long that process takes will depend on how many items are included in the search results.

Once you start using the SCCT features, you can visit **Lists | Supply Chain | Control Tower Dashboard** to see alerts related to your supply chain, a few **Key Performance Indicators (KPIs)**, and the **Predicted Risks** portlet. This is what that dashboard looks like:

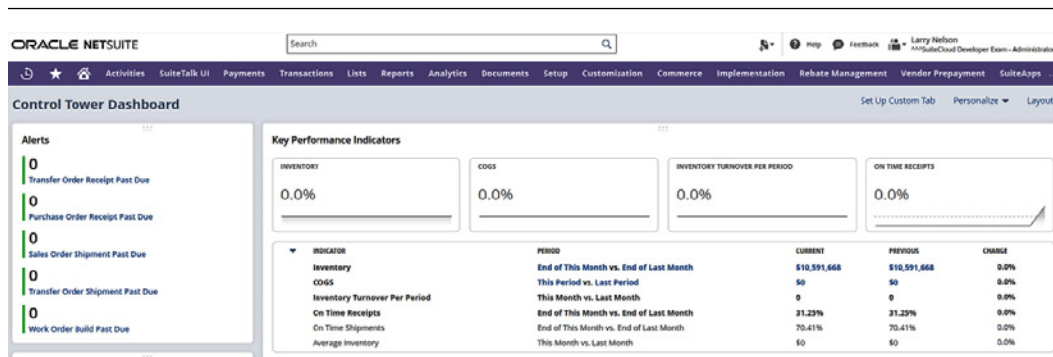


Figure 11.6 – The Supply Chain Control Tower dashboard

NetSuite will manage the alerts for you, based on your items' supply and demand, when any items on purchase orders are past due for receipts or sales orders are past due for shipment. The KPIs help you see a number of things related to the business' supply and demand, so your users can anticipate problems before they occur.

There are quite a lot of these features, so it is worth getting familiar with them yourself before you demonstrate them to your clients. Check out the Help pages under *Supply Chain* and try creating snapshots and using the SCCT dashboard in your test account to learn more.

Setting up items, in all their various forms, takes time to do right, but it is so key to making the rest of the implementation project go smoothly. At this point, you know how important gathering the requirements for item types and their related lists is in the earlier stage of the project. This is because you cannot define these things now without a clear idea of all item uses. You can always come back and repeat any of the preceding steps when new information comes to light – and you most likely will need to – but getting things set up right the first time is still a big help to everyone working on the implementation.

Summary

We typically spend anywhere from two weeks to a month working with a client, getting their item list set up in the system initially. We then tweak the item definitions throughout the rest of the implementation, right up until we go live. Most of the details you set initially can be changed later, but some are tougher to modify, such as the item's type. Following this book's order of events, you should have clear requirements documented before you start to define items in an account. However, we are always open to changing our plans as our understanding of the client's business changes over time.

This chapter should make it clear that the order in which we perform setup steps in an account is very important, and you should note that we need to have a good, solid first pass on the full item list that is imported into an account before we tackle setting up and processing transactions. We will start that work in *Chapter 13*, right after we spend some time setting up the customers and other entity records in *Chapter 12*.

Self-assessment

Here are some challenges for you to consider, which I hope will make you think about how you might handle situations like these within your implementation projects:

1. Your client tells you that in addition to all the products they sell, they also need to offer their customers a way to download user manuals in PDF form for some items. What is one way you can use a native NetSuite feature to meet this demand?
2. Your client likes the feature that allows them to define a new item record in a pop-up window, such as for new discounts they enter on the fly, but they want to control the list of fields available in that pop-up. Is there a way you can set up the item form for just this use case?
3. The warehouse manager needs to be able to directly edit item record details such as the stocking units or preferred bin. But they are complaining about having so many fields on the screen that are not related to their work. How can you help them with this?
4. One of your clients wants to be able to sell items that they purchase from a supplier, and that are then always directly shipped to their customers. Since they never have possession of these items, they probably should not create them as inventory items. How should they set them up in their account?
5. You show your client how matrix items work in NetSuite, and they are put off by the extra steps involved in creating and using them on transactions. They need to track a small list of combinations of color and size for just a few items. Is there any way to use **Item options** for this requirement?

12

Customers, Vendors, Contacts, and Other Entities

In NetSuite, customers place orders, have support cases, and much more. Vendors are whom you purchase goods and services from. We will dig into partners and competitors more than we did in *Chapter 7*, now that you know your client's requirements for these entities. Your job will be to help the client implement all of these entities in a consistent, clear way, and you will learn more about how to do that here.

In this chapter, we will cover the following topics:

- How to set up customers in the account
- Do you need contacts, and should you create them?
- Defining vendors for positive PTP processes
- Setting up competitors, partners, and other entities
- Managing projects and resources

When performing the steps in this chapter with a client, you will be helping the SMEs with customer and supplier relationships, and you will likely interact with the Sales team if your client tracks competitors or partners. Your client will learn how to manage their customers and contacts successfully.

It is a good idea to have read and performed the setup steps described in *Chapter 7* before you try to tackle the topics here with a client during an implementation.

How to set up customers in the account

With a typical client’s implementation, setting up the customers should be straightforward, but getting this right is critical to the success of the project overall. In *Chapter 7*, we gathered the client’s requirements for these entities. With those specific requirements in mind now, we need to help the client start defining their customer records correctly so that they can be used for activities, transactions, support cases, and so on.

This is what the default, new **Customer** screen looks like using the new Redwood theme in 2024.2 before configuration and customization:

The screenshot shows the 'Customer' form in the NetSuite Redwood theme. At the top, there's a header bar with the 'Customer' title, a search icon, and buttons for 'List', 'Search', and 'Customize'. Below the header, there are 'Save' and 'Cancel' buttons. The form is divided into two main sections: 'Primary Information' and 'Email | Phone | Address'. The 'Primary Information' section includes fields for 'Custom Form' (set to 'Standard Customer Form'), 'Customer ID' (with an 'Auto' checkbox), 'Status' (set to 'CUSTOMER-Closed Won'), 'Sales Rep' (set to 'Larry Nelson'), 'Category', 'Default Order Priority', 'Comments', 'Type' (with radio buttons for 'Company' and 'Individual'), 'Company Name', 'Parent Company' (with a dropdown), and 'Web Address'. The 'Email | Phone | Address' section includes fields for 'Email', 'Phone', 'Alt. Phone', 'Fax', and 'Address'. A 'Map' button is located at the bottom right of the 'Address' field.

Figure 12.1 – A new Customer screen in a new NetSuite account using the new Redwood theme

In *Figure 12.1*, you can see the basic fields are available for defining the company or person, including their name, category, and contact details. In a OneWorld account, we also assign each customer to a primary subsidiary, so the system knows which legal entity to default their transactions to (but if you enable the Multi-Subsidiary feature, you’re not limited to this one entity later on).

I want to say how important setting up the default **Customer** form well is to your client’s success, since a business’s customers (and their needs and wants) should always be central to its operations. How can a business be successful without happy customers? As you are setting up these screens and the data in the account, ensure you are enabling your client’s customer service reps to help people quickly and efficiently. When I am a customer of a company, I like it when they can answer any questions I call in with or fix any issues quickly and professionally. If I have to wait on the phone for a long time for

them *to find me in their system*, or I hear them say, “Please wait – the system is running slow today”, I get disappointed or frustrated.

You can help your client avoid this by organizing the customer information into clearly defined blocks users can easily find; do not leave a mix of fields all over the place. We should not let the **Customer** screen become a complicated, unorganized mess of fields, lists, and other elements (some of which nobody even knows whether they are still using).

Do use subtabs and sublists to put sets of fields where people can find them. To help with organizing the screen, I like to watch users work with short, focused tests in NetSuite, to make sure there is as little friction as possible as they click around, doing their jobs. This can take a few iterations, but the improvements we can make at this stage pay off in the long run with happier, more productive users and happier customers.

We typically work initially in the UI, helping the client get the first few customer records entered manually, and working through any issues that come to light. (There should not be many, since you have talked about customers before, but you never know, and many times, issues only come up when we get to this data entry stage.) Once those first few customers are created and the client has a clear idea of how entering them works, we can turn our attention to using the **CSV Import** feature to get the rest of the customers into the system. We do not need this to be a 100% perfect, complete import at this stage, but we do want to show the client that they need to take this seriously and work on really verifying and cleaning up their data for this first import. (Data for lists like *Customers* is frequently messier than we expect it to be.) Later, they can make tweaks and perform updates for any issues they find as they work with the data in NetSuite, but we still try to get it as close to perfect upon that initial import as we can.

To successfully import customer data, the client must be able to export their current customer data from their legacy system. We typically provide a CSV template for this exercise, with headers matching the client’s customer records (and forms, which should already be configured to meet their particular needs). The cleaner the data is, the more likely the client will be to complete this task. Of course, this is not always a simple project. First, most clients will need to determine the right set of filters to apply to their current customer list, to make sure they are not trying to transfer a lot of bad records over from the current system into NetSuite. Then, even valid customers can have invalid data associated with them, such as old addresses or phone numbers, and more. Also note, we typically use the **External ID** field on the NetSuite customer record to give each record a unique ID, especially if it will be used with transactions or other records later. It should be your client’s job to clean up this data since your team will not know good data from bad, and the client needs to own the responsibility of getting this data into their new ERP system in a usable state. They will live with it once they go live, so they need to own this (with your help, of course).

Here are a few general things we typically run into when getting customers imported into NetSuite. This will help you be prepared to assist the users assigned to this task.

Duplicate records

It is very common for most ERP or CRM systems, including NetSuite, to end up with duplicated customer records after being in use for a long time. You probably already know that NetSuite has built-in features that should help mitigate this situation, but not every other application out there does.

When you are assisting a client with their customer imports, make sure to address this and try to remove duplicates from the source data before it is imported. This is always faster and simpler than letting NetSuite do the hard work, and, as I mentioned earlier, this should be the client's job.

One other thing the client should be looking for is the opportunity to define just one customer record in NetSuite where they had multiple variations in their legacy ERP/CRM system. For instance, some other systems can only store one address per customer or can only assign each to one legal entity (subsidiary). Since NetSuite allows multiple addresses and subsidiaries per customer, see if you can help them adjust their customer list down to just what's needed for their new system. (See the *Multiple subsidiaries* section for a bit more on allowing customers to transact with more than one legal entity.)

Bad addresses/phone numbers

Customer records in applications such as QuickBooks or other CRM systems will invariably include some bad addresses and/or phone numbers. Many people belong to very mobile societies, so this is unavoidable. If possible, work with your client to try to apply validations for these data points to their customer import files, again before they are imported. However, some clients will prefer to rely on third-party contact validation tools within their accounts since they are typically used before and after going live. If your project will use something like this, just add some time to the schedule to allow your client to learn how to use SuiteApp and for corrections to be made to the data, once it has been imported. Just ensure there is some sort of bulk validation possible in that case, since the client will not want to manually click a button on every customer they import.

Records that are otherwise invalid

Like any large-scale data import into any computer system, the source data might have records that you or your client will consider to be invalid. This could be because of some category they apply to each customer, or because of some custom data they need to associate with them instead. For instance, we have worked with clients who needed to associate each customer's record with custom pricing groups and amounts. You can try to get this data right during the initial import if that's easy enough, but generally, we work on this as an update or a follow-on activity. If that custom data relies on a custom script or workflow, you may have to wait for those things to be developed before you can get the data right on all of the customer records anyway. *Chapter 18* has more on how we factor customizations and automations into the implementation project's timeline.

Once you've got a plan for importing basic customer data, you can start to dig into some of the more advanced features of the customer record. For instance, NetSuite OneWorld now supports the option to assign a customer to more than one subsidiary. With this, the system still defaults transactions for this customer to use their primary subsidiary, but we can now select another option from a list.

This makes dealing with your client's multiple lines of business much more straightforward. A customer record can also be assigned to multiple currencies, for clients and regions where more than one currency is commonly used. A couple of other features worth noting here are customer-specific item pricing and credit terms. They're both found on the **Financials** subtab and allow us to set sales prices by customer and item, when needed, or control how the customer's credit will be applied by the system.

In addition to getting customers into the NetSuite account, we must also spend time working with the users on how they will maintain them after going live. One really helpful tool for this is the **Customer Dashboard** screen. Here is an example of this feature:

Customer (default)

Customer: 6 ABC Customer Co
 Type: Company
 Company Name: ABC Customer Co
 Status: CUSTOMER-Closed Won
 Sales Rep:
 Partner:
 Web Address:
 Default Order Priority:
 Email: abccustomer@co.com
 Phone:
 Address:
 Primary Subsidiary: Parent Company
 Lead Source:
 Search Engine:
 Keywords (1st Visit):
 Balance: 252.90
 Overdue Balance: 252.90
 Days Overdue: 33
 Date Created: 6/25/2021 7:03 am
 Default Receivables Account: Use System Preference
 Deferred Revenue Account for Revenue Reclassification:
 [View] [Edit]

Key Performance Indicators

INDICATOR	PERIOD	CURRENT	PREVIOUS	CHANGE
Sales	This Fiscal Year vs. Last Fiscal Year	\$608	\$0	↑ N/A
Forecast	This Fiscal Year	\$2,838		
Total Pipeline (Projected)	Current	\$0		
Balance	Today vs. Same Day Last Month	\$253	\$253	0.0%
Unbilled Orders	Today vs. Same Day Last Month	\$2,230	\$1,624	↑ 37.3%
Overdue Balance	Current	\$253		
Days Overdue	Current	33		
Average Days to Pay	Previous One Year vs. One Year Before Last	0	0	0.0%
Average Days Overdue	Previous One Year vs. One Year Before Last	33,568	0	↑ N/A
New Cases	This Fiscal Year vs. Last Fiscal Year	1	0	↑ N/A
Open Cases	End of This Fiscal Year vs. End of Last Fiscal Year	1	0	↑ N/A
Cases Closed	This Fiscal Year vs. Last Fiscal Year	0	0	0.0%
Cases Escalated	This Fiscal Year vs. Last Fiscal Year	0	0	0.0%

Customer Dashboard Links

Sales Reports	Financial Reports	Activities
Sales by Customer Detail	A/R Aging Detail	Tasks
Sales Orders by Customer Detail	Unbilled Cost by Customer Detail	Phone Calls

Figure 12.2 – The default Customer Dashboard screen

This screen is highly configurable, as you would expect, so we should take the time to get SMEs started on this process by removing things they will not need and adding anything new/custom we know will be useful for a typical customer service rep. This might include things such as a search showing their most recent orders, open support cases, or something custom, such as the list of items they have purchased with a warranty. In every case, it is worth working with the users to help them see how they can tweak the screen's layout and contents to help them get their work done quickly and efficiently.

We will cover the importing of customers' open and historical transactions in more detail in *Chapter 15*, but for now, just know that we generally encourage clients to try to go live with NetSuite with the smallest set of migrated customers and transactions they will accept – ideally, that is just the open transactions, and the customers needed to support them.

During this phase, it is very common for clients to ask if they can import all of their historical transactions, but this is generally a bad idea for several reasons. If the client insists on talking about transactions when you are trying to get them to focus only on the customer records, just let them know what your plan looks like, start to discuss it if you have to, and then get them back on track with their entities before too long.

There can also be issues with getting the right list of contacts into the system when you have decided that step is required, so we will cover that in the next section.

Do you need contacts, and should you create them?

As we discussed in *Chapter 7*, customers can be created for an individual or a business, and each customer record in NetSuite can have one or more contacts associated with it. The main reasons for using **contacts** are when you need to differentiate people who work for the customer (billing versus shipping people versus executives), or when you need to give only some of the company's users the right to log in, either to the NetSuite **Customer Center** or to a **SuiteCommerce** web store. You can always use contacts even when this is not a requirement, but generally, we leave it to the client to decide how they want to use records.

Once the decision has been made to use contacts, getting them into the system should be straightforward. We do not generally customize records, so you can provide a generic CSV import template to the users responsible for this. They can import customers and contact records into the system in the same step if they want to, but it is usually simpler to make two separate lists, so you should make two separate imports. Of course, if the client's source data (from their legacy system) contains the customers and contacts combined, it might make more sense to try a single, combined import instead.

One thing to do to make sure that the client is clear on with contacts is the extra data maintenance. Generally, we like to make sure the client keeps their contacts as light as possible, in terms of the data we store. We do this for several reasons, including their own time and effort, and the various data privacy laws in various regions that dictate what data they store, for how long, and so on. Many clients start with the idea of *Let us get and keep everything we can*, but that is no longer wise. Data storage is cheap with NetSuite, but making sure the business is compliant with the laws and regulations that apply to it is more important.

Once the first large (but not necessarily full) customer and contact imports are made, you and your client can start working on vendors.

Defining vendors for positive PTP processes

Getting the list of vendors into NetSuite should be simple, and just as we did with customers, we usually create one or two manually and then transition to working on a CSV import. We help the client pull the data from their legacy ERP system and set up a CSV import job to help them with mapping the fields and lists of related data.

Since vendors are used within the various Procure-to-Pay (PTP) transactions, we need to pay attention to a few things at this stage to avoid issues later on.

Multiple subsidiaries

In a OneWorld account, among a set of other useful features, we have the ability to use multiple subsidiaries. (See *Chapter 1* for a refresher on OneWorld.) When setting up vendors, think carefully about which subsidiaries each vendor should be associated with. By enabling a vendor to work with more than their primary subsidiary, NetSuite allows us to select which subsidiary will be used on each transaction (for example, **Purchase Order**) we create for that vendor. If your client wants to allow both their *School Supplies* and *Sporting Goods* subsidiaries to order equipment from *ABC Sports Co.*, then this is fine, but this will have an impact on accounting and reporting going forward. Read the *Subsidiaries* section of the Help page titled *Add a Vendor Record* for more information.

Addresses

Just as with customer records, vendors can have multiple addresses, and we can choose which to tag as the default for shipping versus billing. The address you mark as the default billing address will also be used when creating payments for the vendor, so think of that as the default payment address as well.

Financial considerations

On the **Financial** subtab, we can make a set of important choices for things such as terms, tax status, and more. Here is an example of these fields in a new account:

Subsidiaries	Relationships	Communication	Address	Financial	Preferences	System Information	Time Tracking
Account Information							
LEGAL NAME ABC Supplies				DEFAULT VENDOR PAYMENT ACCOUNT Checking			
ACCOUNT 				PRIMARY CURRENCY ★ USA			
DEFAULT EXPENSE ACCOUNT Miscellaneous Expense				TERMS Net 30			
DEFAULT PAYABLES ACCOUNT Accounts Payable				CREDIT LIMIT 1,000,000.00 (USD)			
				INCOTERM 			
Tax Information							
TAX ID F34587GA234				<input type="checkbox"/> 1099 ELIGIBLE			
Balance Information							
BALANCE 1,972.00 (USD)				UNBILLED ORDERS 0.00 (USD)			
PREPAYMENT BALANCE 0.00 (USD)							

Figure 12.3 – The default Financial tab for a new vendor record

Set **ACCOUNT**, **PRIMARY CURRENCY**, **TERMS**, and **INCOTERM** as you need for each vendor. For companies doing business in the US, enter their **TAX ID** and select **1099 ELIGIBLE** when that is appropriate. Most NetSuite clients use a third-party package for generating their 1099s for vendors, but NetSuite can help gather the information needed for those documents when this is enabled. See the Help page titled *Vendor Records for 1099 Contractors* for more information on this.

Vendor bill/three-way matching

Some clients will have vendors for whom they need to use either NetSuite's native **VB matching** or a more advanced **3 way match** solution/add-on (more on this in *Chapter 14*). For those vendors, we need to set up the **PO** and **Item Receipt** tolerance fields, or the set of fields provided by SuiteApp. Then, we can use this tool to match these transactions at the subsidiary, item, or vendor level. Most businesses that make purchases will need this, so this is an easy win if they can match their process to the way either NetSuite does out of the box or their chosen SuiteApp solution can support.

If the business does billing before receiving, though, they won't be able to use this feature.

You should have a good handle on setting up vendors for your client now, so as a final topic, we will discuss the other entity types within NetSuite.

Setting up competitors, partners, and other entities

As discussed in *Chapter 7*, these entities are not always needed. When your client does need to track their competitors, though, they will almost certainly need to secure the data, to keep some or most users from accessing it. Use the **roles** you set up previously for this, by removing the competitor list from lower-level users' permissions and making sure only the right people are allowed to edit and view these records.

When a client wants to use **partners**, it is usually so they can enable promotions and similar offers for these selected entities. This is an opportunity to explain to them how the **Partner Center** can be useful in this regard. For example, if a company signs up partners to sell into certain markets, they might offer incentives to those partners for sales completed within certain time frames or for a specific product line. You will also do well to know about other third-party offers from NetSuite's partners to fill in any gaps in this functionality. Check out the SuiteApp marketplace for lists of products that help with things such as loyalty programs, promotions, and more.

NetSuite also offers another handy entity type known as **groups**. These can be defined as either a static list of other entities (customers, contacts, employees, jobs (also known as projects), partners, or vendors) or dynamic groups. A dynamic group is defined based on the results that are returned from a search, so the membership will change over time as you use the system. We typically set up groups for several different reasons, such as for customers who get special pricing, for those we want to send specific marketing communications to, or vendors we want to contact via regularly scheduled messages. We can also set up groups for administrator employees who need to be notified when customization or integration has problems or reports errors. We might also need to set up groups for use with Sales teams or for support purposes, such as case management.

Getting the client through this phase, and thus creating their first sets of customers, vendors, and other entities, is important, so we tend to spend time on it. How much time it takes just depends on how quickly the client's SME users become familiar with the intricacies of CSV imports, as well as how hands-on they will be with fixing form and custom field configuration issues if they arise. This phase is a great opportunity for you to start making your users more and more responsible for things like this, so they can start to feel what owning their own NetSuite account will be like when they go live.

I do not usually have to remind clients that I will not be around for long post the go-live stage, but occasionally, I have to gently encourage them to get more involved when it comes to tasks like those covered in this chapter.



Managing projects and resources

In *Chapter 7*, we talked about how you should come to understand a service-based client's projects and how they staff them with resources. Let us focus here on how to set them up and get them started using these features within their NetSuite account.

A **project** (also known in NetSuite as a **job**) in NetSuite is like a sub-record you can create for any customer. Each customer can have one or more projects, and projects have one or more **resources** assigned to them. For instance, if your client provides cleaning services for senior citizen facilities, they might want to have each facility as a customer and each cleaning service contract as a project. They might assign a manager and a staff of cleaners to each project, so they can keep track of who works at each facility, the start and end date of the contract, and so on. The tasks on the project might include some setup that they will perform, plus each cleaning service they will provide. NetSuite allows us to then keep track of the costs of these resources over time.

There are two ways to set up projects in NetSuite, and they each serve different uses. **Basic projects** are really, really basic; your clients can track a basic project, but they cannot define the tasks that make up the project or assign resources to specific tasks. Most NetSuite clients will need the more advanced option, known as **Project Management**. When this feature is enabled, you can then assign tasks to projects and resources to tasks. There is a really handy table explaining the differences between these two types of projects in NetSuite Help, on the main *Projects* page. The rest of this section assumes you have enabled **Project Management** for the more advanced features it offers.

This is what a new **Project** record looks like in NetSuite with the new Redwood theme:

 **Project** 

ListSearchCustomize

SaveCancel

▼ Primary Information

Job ID ^{*}

Copied From Name

☒ Auto

Project Name

Customer

ABC Marketing Inc

▼

Project Manager

▼

Status

▼

Project Template

▼

Language

English (U.S.)

▼

▼ Project Overview

Estimated Work

Actual Work

Percent Work Complete

Allocated Work

Remaining Work

Percent Complete by Allocated Work

▼ Project Dates

Start Date

3/23/2025

Calculated End Date

Last Baseline Date

▼ Classification

Subsidiary ^{*}

US - West

▼

Figure 12.4 – A job record in NetSuite created with Project Management enabled

Projects can be tied to transactions, including opportunities, quotes, sales orders, and support cases. For example, you might want to sell your cleaning services to a client and so create an opportunity initially recording the client's name and the service items that would be included. You might convert the *opportunity* to a *quote* to get specific prices down and then convert the quote to a *sale* when the client accepts the details. You can have NetSuite create a project based on the sales order automatically if you choose. Once the project has begun, you might like to associate cases with the project, whenever the customer reports an issue with the service being provided. Employees can then create entries and associate their entries with the project to have NetSuite automatically track the costs of the services as they are entered.

If the basic project features are not enough for your client, which might be the case if their entire business is service oriented and there are lots of employees performing the work, then you can consider other options, including adding on the integration with another Oracle offering, **OpenAir**. This is a separate Oracle product, but it is entirely aimed at professional services businesses with lots of employees and can help them manage time and expense entry for lots of projects and includes more robust reporting and management options than NetSuite by itself.

With these steps, you should have all of the entities your client needs set up, at least initially so that they can use them in their transactions, reports, and so on. This is key to the upcoming chapters, so congratulations on a job well done!

New feature highlight – Specialized User: CRM and Read-Only Roles

New in the 2024.2 version, NetSuite now offers two new types of roles for clients. The first, the **Specialized User: CRM role**, is meant to be used as a template for people to manage sales, product configurators, marketing, and so on. With this role, those folks will be able to access customers, sales, support cases, and similar CRM-related records, but will not be able to access Vendors, Procurement, or unrelated other transactions. The roles are customizable to fit each client's needs. The second new type of role is called **View and Approve**. We should use this role as the starting point for users who are managers and others needing limited access to NetSuite. They will be able to perform approvals and to view records they're granted access to but will not be able to edit most records. These roles are also customizable to the extent that you can remove permissions your clients' users won't need (but you can't add any additional permissions). Think about how best you might use these new roles in your next implementation.

Summary

When you are working on a client's implementation, you are helping them learn how to use NetSuite and also helping them configure their account for their specific business needs. Setting up their customers is probably the most important aspect of this, since you need to have all of the customer data easily accessible and viewable, and you want to take advantage of all the NetSuite features that will use this data, saving the users time and effort. Taking the vendors and other entities into account can be more straightforward and can be nearly as important for some clients. Service-oriented companies can take advantage of the project features in NetSuite to make tracking their employees' overtime simple and efficient.

Hopefully, you have seen how we can help the client do all this in many ways, and the longer you work with NetSuite, the more ideas you will have about making the system work for each of your clients.

In the next chapter, we will move on to transactions. We will start with the financial team's work for things such as **Journal Entries** and **Period Closes**.

Self-assessment

Setting up a business' customers and vendors can include a few surprises and challenges, so here are a few scenarios related to these topics for you to imagine how you would solve them:

1. A client asks you how they can control the pricing for a set of customers. They want to charge custom prices for about 1,000 items for just about 20 of their *top* customers. How can you help them achieve this?
2. One of your clients wants to be able to have parent and sub-customers, but they are concerned because this relationship can be more than one level deep. In other words, one parent customer might have a set of sub-customers who each might have their own set of sub-customers. Does NetSuite support this and, if so, how deep can that nesting of accounts go?
3. You are 1 week away from the planned go-live date for a client when the head of their customer support department says to you, "We were not planning to use NetSuite's Support Cases and related features before, but now we are considering this". What is the simplest approach you can find to getting them started on this, without jeopardizing their go-live? (Think about using fields and lists on the customer record only as a first step, maybe.)
4. *Party Central Station* runs parties for other people and needs to tie their customer records into the calendar to track the events the customer has paid for. Without creating a script or a workflow, how can you use native features to help them with this?
5. Sandra is the head of sales for a client and, one day, she lets you know that one of her salespeople has left the company. She needs to replace their name on a few hundred customer records as quickly as possible. What is the best option for doing this?

6. Your client says they want two people to be able to review and approve employee requisitions, but otherwise, those people should not have access to anything else in NetSuite. What's the fastest way to get them set up with just the right permissions in the account?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search for this book by name.

Note: Keep your purchase invoice ready before you start.



Financial Transactions and Period Closes

The bookkeepers, accountants, and financial managers within any organization will learn to love the flexibility that NetSuite provides them to work with journal entries, period closes, taxes, and more. You need to learn how to make sure your client's financial experts and accountants are satisfied so that they will be able to complete their tasks every day, including at the end of their financial periods, with NetSuite.

In this chapter, we will cover the following topics:

- Setting up and using the General Ledger (GL)
- Implementing currency management
- Setting up and using banking features
- Using Journal Entries
- Performing and managing Period Closes
- Overview of NetSuite's tax features

Your client's financial managers, bookkeepers, and accountants will be your main contacts for the topics covered in this chapter. They should have already completed some NetSuite introductory training before you begin training them further on the system's accounting features. After you have walked the client through these topics, they will be able to navigate and use the finance and accounting features themselves.

Setting up and using the GL

As we discussed in *Chapter 6*, most clients will know what they want their Chart of Accounts (COA) to consist of when they transition into the cloud. NetSuite does come with a COA list in every new account, just in case you ever work with a brand-new company, but in most cases, we remove those accounts early in the project and replace them with the company's list. Most established businesses will have accounts they need to import and use right away. We typically help each client import their COA from a CSV file since this is relatively simple to do (and it helps the financial users become familiar with using the **CSV Import** feature). The good news here is that the system offers a wide variety of account types (*expense, income, credit card*, and so on), which should satisfy most business accounting needs.

Remember that when a business has a OneWorld account and uses multiple subsidiaries, everything about its finances and accounting is more complicated. With OneWorld, we gain a number of useful features, including subsidiaries, multiple currencies, and multiple accounting books, if they're needed. Each of the features explained in this chapter works well with OneWorld when needed, but adding them to an account makes everything from transactions to closes to reporting more complex to maintain over time.

The COA can become a topic we spend considerable time on when there are multiple subsidiaries in a OneWorld account – for instance, if the client's business is broken up into multiple main legal entities, and they each have a hierarchy of subsidiaries within them. We can model the list of subsidiaries within one NetSuite account easily enough, but setting up and getting that hierarchy just right, including being able to eliminate subsidiaries and accounts, is key here since all other GL-related activities in the account will depend on this being done correctly. In rare cases, we need to take that a step further and set up multi-book accounting.

We will not have enough space to cover this topic in this book, but you can start to learn more about this within the NetSuite Help pages – go to the *Accounting* heading for more on all of this chapter's topics.

The next step in this process is typical for helping users get used to NetSuite's approach to posting in the GL. Since other applications have different names for some transactions or lack some of NetSuite's transactions, it helps the users know which transactions are *posting transactions* and which are not.

Some accounting users might start an implementation project with the expectation that their sales orders should be posted to the GL, for instance, but a quick explanation of why NetSuite considers these to be non-posting transactions usually sets them straight. The answer is that the items on a sales order might change or be removed before they are fulfilled/billed to the customer. A good rule of thumb we use says that if the transaction can go unprocessed or if the inventory from the transaction might not be affected after all, it shouldn't be posted to the GL. Purchase orders, opportunities, and sales orders are non-posting transactions because they might never be received or fulfilled/sold. Vendor bills, invoices, and inventory adjustments are posted to the GL because they indicate a finished deal with someone else, or inventory levels are affected.

There are many other related NetSuite features within this financial/GL heading, but not every client company will need to use all of them. If your OneWorld client has multiple subsidiaries that do business with each other, you will want to research what NetSuite calls **intercompany** accounting in the NetSuite documentation (search for *Automated Intercompany Management*). For instance, we use intercompany purchase and sales orders to record a purchase by Subsidiary A from Subsidiary B and a sale of the same items from Subsidiary B to Subsidiary A. We need to set up the subsidiaries as vendors and customers to complete those transactions, but it makes the accounting and reporting very clean once we've set it all up. If the business needs to track non-monetary data about the financial reports, they should check out the **NetSuite Statistical Accounts** feature and its possible uses.

For companies that only ever conduct their transactions in one currency, accounting management is relatively simple. Let us explore what it looks like when multiple currencies are at play.

Implementing currency management

NetSuite is available globally now, which means that it must support everything businesses in various regions around the world require. That includes things such as language translations and data storage rules for the EU versus the US, for instance; it also means we need a set of features related to currency management. When a business tracks all of its work in just one currency, it can set that as the default and then use that one currency for all transactions, reports, and so on. But many companies today work in regions where multiple currencies are required, and NetSuite has them covered – for the most part.

The **Multiple Currencies** feature can be enabled by going to **Setup | Company | Enable Features**. At this point, the business must select a *base currency* (for each subsidiary where applicable) to act as their default (that is, most common) currency for transactions and reports. Every other currency they enable will be referred to as a *foreign currency*.

For example, if Ottawa Soaps is a manufacturer in Canada, they might need to buy and sell things in both CAD and USD (Canadian and US dollars). CAD will be their base currency and USD will be a foreign currency for them. Note that NetSuite sets the default for the base currency based on the country the business is primarily set up in, and you can only change that value before any transactions (of any type) are created in a new account. Once a base currency is set up for each subsidiary, NetSuite will use that for financial reports as well.

Once you have established the currencies that will be used within the account, you can start applying those currencies to entities such as customers and vendors, and to every transaction with currency amounts (for example, **Item Fulfillment** does not have any currency-specific amount fields).

In addition to helping the client get this set up, there are a few other topics you should cover with a client who has this requirement. We will check them out in the following sections.

Customer and vendor currency lists

Each customer and vendor record in the system can be associated with one or more of the account's enabled currencies. In most cases, most clients are OK to set up the list in advance for their accounts and then allow all customers and all vendors to use that list. If the client wants, though, they can remove entries from the currencies list on a per-customer/vendor basis. For instance, maybe your client's company has a US business and a *worldwide* business, and the US customers are only allowed to transact in USD, but the worldwide customers can use USD, euros, or pesos.

This is what the **Currencies** list of a typical customer record might look like:

Currencies • Credit Cards Group Pricing Item Pricing Time Tracking					
CURRENCY *	BALANCE	DEPOSIT BALANCE	OVERDUE BALANCE	UNBILLED ORDERS	FORMAT
USA	0.00	0.00	0.00	0.00	Format Sample: \$1,234.56 Symbol: \$ Symbol Placement: Before Number
Euro					Format Sample: €1 234,56 Symbol: € Symbol Placement: Before Number
Canadian Dollar					Format Sample: \$1,234.56 Symbol: \$ Symbol Placement: Before Number
<input type="text"/>					
<input type="button" value="Add"/> <input type="button" value="Cancel"/> <input type="button" value="Remove"/>					

Figure 13.1 – The Currencies list, on a customer, in the Financials subtab

We can add currencies to this list and remove them as needed, on a per-customer or per-vendor basis. NetSuite automatically tracks the **BALANCE** and **UNBILLED ORDERS** amounts in those lists for us.

Exchange rates for multiple currencies

Once a company starts tracking its transactions using more than one currency, it usually becomes important to plan how it will handle exchange rates between those currencies. For reporting reasons, the business needs to know how much they bought and sold in any period, and it is usually easiest to do this by converting those so-called *foreign currencies* into the account's *base currency*. Some reports are defined to show both the foreign currency amount and the converted base currency amount, but this all relies on a well-managed exchange rate solution within the account.

Out of the box, the system has a screen we can visit to set up the initial exchange rates when we are first starting. Go to **Lists | Accounting | Exchange Rates**. You will see the currencies that are enabled there and a row in the list showing the exchange rate for converting from each of the foreign currencies into the account's base currency. You can set up new rows in this list whenever you need, for instance, to define the exchange rate for a new currency you just set up, but the base currency is always chosen first for any exchange rate entry. We can set an effective date at this time as well.

Here's what the currency exchange rate might look like in a NetSuite account:

Currency Exchange Rates


History


New

FILTERS

TOTAL: 36					
BASE CURRENCY	SOURCE CURRENCY	EXCHANGE RATE	EFFECTIVE DATE ▼	RATE PROVIDER	METHOD
British pound	Euro	0.9026	03/01/2025		Manual
British pound	US Dollar	0.8023	03/01/2025		Manual
Canadian Dollar	British pound	1.6941	03/01/2025		Manual
Canadian Dollar	Euro	1.5293	03/01/2025		Manual
Canadian Dollar	US Dollar	1.3591	03/01/2025		Manual
US Dollar	British pound	1.2463	03/01/2025		Manual
US Dollar	Euro	1.1251	03/01/2025		Manual
US Dollar	Canadian Dollar	1.559	09/01/1986		N/A
US Dollar	US Dollar	1	09/01/1986		N/A

Figure 13.2 – An example of a currency exchange rate in NetSuite

 **Quick tip:** Need to see a high-resolution version of this image? Open this book in the next-gen Packt Reader or view it in the PDF/ePub copy.

 The next-gen Packt Reader and a free PDF/ePub copy of this book are included with your purchase. Scan the QR code OR go to packtpub.com/unlock, then use the search bar to find this book by name. Double-check the edition shown to make sure you get the right one.



As part of getting the client live, though, they will most likely want to enable an exchange rate syncing service, such as the one NetSuite offers. To enable this, visit the **Enable Features** screen, and then the **Company** tab. When enabling the **Exchange Rate Integration** feature, you will need to visit the **Accounting Preferences** screen (**Setup** | **Accounting** | **Preferences**) to select from the list of available exchange integration providers. NetSuite contracts with these companies, and the list can change over time.

Currency reporting is another level of complexity we learn to handle when we use the currency features in an account. Clients with OneWorld accounts might like the built-in **Consolidated Exchange Rates** feature, for instance, which makes it easier to manage exchange rates across subsidiaries for each accounting period. When you visit **Lists | Accounting | Consolidated Exchange Rates** in one of these accounts, you'll see the list of periods and subsidiaries and the current, average, and historical exchange rates NetSuite is using for each. Some clients may also require assistance with groups of users and with formatting for numbers and currency symbols. We use the **General Preferences** and **Set Preferences** screens for this. Some accountants will have *interesting* ideas about how their reports need to arrange the data or which numbers to show from each transaction (the base currency or foreign currency amounts, for instance). Just work with them and the features of the system to accommodate each request.

Next up, let us talk about the banking features NetSuite provides out of the box, since this is always an important topic for the accounting users we work with.

Setting up and using banking features

NetSuite offers a variety of features for banking (deposits, checking, credit card handling, and reconciliations). Many of those features can be automated in some sense (depending on the countries involved) via imports and exports.

Always remember that there are special permissions for most of the banking features, so unless you access NetSuite with the administrator role (which only a few people should do), you will need someone to grant your role these permissions before you can access the features. Most of the time, a given user will be assigned the role they need to use these features, but it's still a good idea to read the Help page titled *Permissions for Banking Features* for more information.

Your job in helping the client set up these features will be to get to know them well and to know how to tweak them if that is needed. For instance, features such as **Deposits** and **Write Checks** are easy enough to learn about, but your client might want to adjust the deposit slip or check the templates the system uses when printing those things (if they still print them on paper, of course). The automated payment and reconciliation and bank feed features all require considerable technical assistance to set up correctly. There are a lot of bank data formats in use today. Selecting a format from the list NetSuite offers is a good start, but we frequently need to make tweaks to formats to get to 100% compatibility. Or your client may work with a financial institution that NetSuite does not already know about; in those cases, you will need to set up a bank file format from a template, and the related data, from scratch. See the Help pages on *Financial Institution Records* and *Format Profile Creation* for more information.

The system has a good number of features built into it for handling basic bank data, but you can also check out SuiteApp Marketplace in any account, or visit suiteapp.com to learn about and install tools from NetSuite for bank statement or feed imports. There are third-party tools there as well for companies outside the US with region-specific banking requirements or situations that require other, more advanced features.

Once we have the basics in place for the business's COA, currencies, and banking, we can move on to working with the accounting team on how they will use journal entries throughout their business, when they need to make adjustments to GL balances directly.

Using Journal Entries

There are many reasons why we end up with entries in the GL within an active NetSuite account. Some of those are user-initiated and some are system-generated – for instance, users create GL-posting transactions such as invoices, payments, and journal entries, and in some companies, users will need to create **Advanced Intercompany Journals (AIJs)** as well. The system auto-generates a variety of journals as well, for things such as **Fixed Assets Management** (also known as **FAM**; see the following feature highlight section) and **revenue recognition** (most commonly shortened to **rev rec**). In most cases, we want the accounting team to know about the system-generated journals so that they can anticipate their impact on the business (and tweak them in the rare cases that this is needed). When a client requires a custom solution for things such as journals or the GL impact of other transactions (via SuiteScript), we want to be very careful both when delivering such customization (with ample testing, for instance) and afterward, as we also need someone to actively monitor the customization once it is in place.

Another handy OneWorld feature to know about is NetSuite's **Expense Allocation Schedules**. By setting up a set of schedules for your client, you can simplify their management of expense GL impacts within each subsidiary. These allocation schedules aren't used for intercompany adjustments, but they do allow you to mix allocations across departments, locations, classes, and custom segments. For instance, if a business wants to split up their rent for a location across departments, they could set up an allocation schedule for this purpose and so we won't have to create a complex journal manually each month.

I have been involved in many projects where we developed and tested a solution that customized the GL impact of transactions or posted new journals to change GL balances as the customer required, and in one of those cases, the customer reported much later that the scripts had caused the wrong accounts to be debited and credited for the wrong amounts.

Always remember that messing up a sales order is bad (the customer will not be happy if they do not receive the right items, for instance), but messing up thousands of journals could conceivably ruin a business completely if they cannot complete an audit, or worse. If you are the functional consultant on a project, make sure you either have the right experience to handle a task like this or work with someone who does.



Let us look at some uses I have come across for (user- and system-generated) journal entries in the system that you might not already be aware of or that might just require some additional explanation.

Feature highlight – Fixed Assets Management

Whenever your client’s business involves depreciation for things like buildings, vehicles, and equipment, you will most likely need a solution that covers their tracking and reporting needs. NetSuite offers an FAM solution as an ERP add-on for automating most of the tedious and repetitive parts of this process. By providing records, fields, and automations for FAM, the system makes accountants and financial professionals’ work much easier. With the NetSuite bundle installed, they will define all the details around their assets (including costs and depreciation schedules) and then manage them over their lifespan. This feature includes everything most businesses will need to track leased items as well. Read the Help pages under *Accounting | Fixed Assets Management* for more details on this useful feature.

Reversals

There will be times in many businesses when a journal is created for one GL impact, but you must reverse that impact later. You can handle this with a standard Journal Entry transaction by entering a reversal date on the forms. These reversals are automatically linked to the original journal they reverse. Here is an example of a Reversal Journal:

 **Journal** 

Primary Information

ENTRY NO. To Be Generated	DEBIT	CREDIT	DATE * 7/30/2021
OUT OF BALANCE BY			POSTING PERIOD Jul 2021
CURRENCY * USA			REVERSAL DATE 7/30/2021 <input type="checkbox"/> DEFER ENTRY
EXCHANGE RATE * 1.00			MEMO

Classification

SUBSIDIARY *
PRIES Sporting Goods

Lines

0.00 •

ACCOUNT *	DEBIT	CREDIT	MEMO	NAME
Advances Paid		100.00		
Accounts Receivable	100.00			

Figure 13.3 – Entering a reversal date into a journal

The system defaults the reversal date on these transactions to the current date, but you can change that if you need to. That **DATE** field is the key to making a journal act as a reversal.

Advanced versus normal intercompany journals

When our client's business has more than one subsidiary and they buy and/or sell goods or services between them, we frequently use **Intercompany Journal Entries (ICJs)** to record the details. Before NetSuite version 2017.1, we just had ICJs for this, but then AIJs were introduced. The advanced version is now the default, and the non-advanced version is available only as a legacy feature. The AIJ transaction allows us to specify both a *from* subsidiary and one or more *to* subsidiaries, allowing for more complex transactions when they are needed. It's a good idea to review the Help system pages for these more advanced transactions before you help a client learn to use them.

Here is an example of an AIJ entry. You can see that it's *advanced* because it has the **SUBSIDIARY** list field on the line items:

Advanced Intercompany Journal

Save Cancel Actions

Primary Information

ENTRY NO.
To Be Generated

DEBIT CREDIT

OUT OF BALANCE BY

CURRENCY *
USA

DATE *
4/23/2023

POSTING PERIOD
Apr 2023

REVERSAL DATE

☐ DEFER ENTRY

MEMO

Classification

SUBSIDIARY *
PRIES Sporting Goods

Lines Communication

0.00

Clear All Lines

SUBSIDIARY *	ACCOUNT *	DEBIT	CREDIT	MEMO	NAME	DEPARTMENT	CLASS	LOCATION	REVENUE RECOGNITION RULE	SCHEDULE
⋮ PRIES Sporting Goods	Accounts Receivable	1,001.00								
⋮ Parent Company	Employee Advances		1,001.00							

Figure 13.4 – An example of an AIJ in NetSuite

Revenue recognition


Companies such as software and services companies frequently need to control when and how they recognize their revenue since they do not primarily deal with goods in physical form.

There are complex rules for accounting around rev rec in each country, but the system enables them via its **Advanced Revenue Management (ARM)** module. This set of features will automatically include system-generated journals to record and adjust revenue based on the transactions being entered and their details. Setting up and training users on this module requires additional skills and experience, but you can learn more about this via the ARM Help pages in the system.

Elimination

Businesses with parent companies and subsidiaries frequently need to track the profit or loss involved in doing business between these entities. NetSuite makes this relatively easy with elimination journal entries. This is typically very important for these companies since, without these journals, their financial reports will not be correct and revenue/expenses might otherwise be double-counted when looking at consolidated financial reports. An elimination journal in the application is the same transaction we use for other purposes, but it is associated with an **elimination subsidiary**.

Here is an example of a simple subsidiary hierarchy showing how **Elimination Subsidiary** fits in:

 **Subsidiaries**

List Search Audit Trail Subsidiary Settings Manager

VIEW

Subsidiary Default






Customize View

New Subsidiary

FILTERS

STYLE

Normal

     SHOW INACTIVES

TOTAL: 3

EDIT VIEW	INTERNAL ID	NAME ▲	ELIMINATION
Edit View	1	Parent Company	No
Edit View	3	Elimination Subsidiary	Yes
Edit View	2	PRIES Sporting Goods	No

Figure 13.5 – A typical subsidiary hierarchy with a parent company

As you can see in *Figure 13.5*, placing **Elimination Subsidiary** in this position, below **Parent Company**, allows the users to create elimination journals for any of the subsidiaries at the same level or lower in the hierarchy.

Statistical journals

All the previous journals record monetary values the business needs to track as it goes about its day-to-day or month-end processes, but there might be times when your client needs to record journals for amounts other than money. For example, if your client provides cloud-based computing services and wants to track the total number of terabytes of storage offered, they can do so with statistical journals. Once this feature has been enabled in an account, these transactions can be created in the same way we create financial journals, except we track a specified unit of measure (for example, terabytes or

square feet of floor space) in the journals instead of currency amounts. These records can be edited, deleted, approved, reversed, and imported as needed.

Once your client is comfortable with using the various types of journals within the system, you can help them plan and practice performing their period closes. We will check this out next.

Performing and managing period closes

A **period close** in NetSuite is a flexible thing, and many companies find it is a little too loose out of the box. However, this depends on the business and what they are used to from their legacy ERP system. Some companies need to keep transactions editable from one period to the next in case working conditions change over time. These companies will do well to use the period close feature's minimal settings.

However, many other businesses would be horrified at the thought of allowing users to edit transactions in previous accounting periods, so they need to have the system lock down each period as they close it to keep their reporting accurate over time. These NetSuite clients will need to understand how the period close feature can be used to lock their transactions and reports in place. They almost always already have a process in place and policies for this, so now they just need to understand how to implement those policies with NetSuite's various features. The **Period Close Checklist** screen is a very handy feature for users who will perform these activities in their accounts.

The checklist is built by the system based on the features you've enabled in the account and can't be customized.

This is what the native checklist screen looks like for closing a period:

Period Close Checklist: Jan 2025

PERIOD NAME Jan 2025		START DATE 01/01/2025
STATUS Open		END DATE 01/31/2025

Notes

GO TO TASK	TASK	STATUS	MODIFIED BY
	Lock A/R	<input type="radio"/>	
	Lock A/P	<input type="radio"/>	
	Lock All	<input type="radio"/>	
	Resolve Date/Period Mismatches	<input type="radio"/>	
	Review Item Line/Inventory Detail Quantity Mismatch	<input type="radio"/>	
	Review Negative Inventory	<input type="radio"/>	
	Review Inventory Cost Accounting	<input type="radio"/>	
	Review Inventory Activity	<input type="radio"/>	
	Review Custom GL Plug-in Executions	<input type="radio"/>	
	Generate Intercompany Cross Charges	<input type="radio"/>	

Figure 13.6 – The Period Close Checklist screen, partially completed

In the checklist, you can see that NetSuite will try to help the accountants with the steps in the close, including things such as resolving date/period mismatches and consolidating exchange rate calculations.

You should have already set up periods in the account that match as closely as possible those that the business currently uses. In a brand-new account, you will find that the system has already created 12 periods for the current year, one per month. We can replace those periods with any of the supported alternative options. For instance, a client might want to use the *4-4-5 weeks* option to change the duration of their periods. We can always create new periods for the future, but we can only replace the current periods when no transactions have been assigned to the period yet.

For businesses using the period close feature to lock their account down within each period, you can help them understand the additional permissions that affect closed periods. Let's review each of those permissions.

New in 2025 – NetSuite Account Reconciliation

Just introduced in 2025, NetSuite now offers a more robust and feature-packed way for organizations to perform their period closes. **NetSuite Account Reconciliation (NSAR)** is available for businesses with more complex close requirements. For instance, the more GL accounts and bank accounts a business has to reconcile each month, the harder this job becomes. Most accounting teams spend significant effort closing their books each month (or however often they do this), and so NetSuite offers NSAR as a way to speed up and automate this process.

With this integration added into a client's NetSuite account, they can export the relevant accounting and banking details to the Oracle Enterprise Performance Management system (an external system offering from Oracle) and then analyze and track their closes better than native NetSuite can by itself. NetSuite Planning and Budgeting is another Oracle tie-in that can help NetSuite clients improve their bottom line and gain efficiencies at the same time. This page in the NetSuite documentation covers these topics and more: https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/book_5113648189.html.

Manage accounting period

This permission allows the user to work with accounting periods, create them, change them, and so on. This is separate from the permission needed to perform period closes.

Override period restrictions

Closed periods can be reopened (in rare cases, only when necessary) by any user with this special permission associated with their role. This permission also allows the user with this role to add and make GL-impacting changes to posting transactions in periods that have been locked for transactions. For example, if the business closes the April 2020 period but then realizes it incorrectly reported a liability, it might want a senior accountant to make a change to a transaction to correct this. A user can only do this if they have the **Override** permission. This can be key for certain users who very rarely need to make changes that would otherwise be locked down in their accounts.

Period closes within the system can be performed on any day, and that should work well with most of your clients' schedules. A typical accounting department does not close the previous month on the first day of the following month, for instance. Learn how to use the close features yourself and then work with the accountants to practice closes at least a couple of times during the implementation, before going live, to make sure there will be no issues after that date.

Most accountants believe a successful period close is the most important thing they can do in the system. Getting the financial reports to be 100% accurate and complete is critical if the company is publicly traded, but this is very important to privately held companies too. Practicing this with these users and getting any issues handled before going live is yet another key to a project's success.

Overview of NetSuite's tax features

Taxes in NetSuite are almost universally used. How each company uses the native tax features varies based on a number of factors: where your client is based (every country has its own tax regulations), what kind of business they run (some companies' sales are entirely tax-free), and more. Here, I will provide an initial idea of the native sales tax features NetSuite provides but know that there is a lot more to this topic. For a business, calculating tax amounts correctly is both pretty complicated and very important, so it is critical that we help them set up and learn to use the tax features during implementation.

Generally, most businesses in most countries will need to collect taxes on at least some of their sales, they probably pay taxes on some of their purchases, and they might pay taxes on some of their employees as well. NetSuite's core product offers a list of features that should help clients get started, and they also offer a variety of pre-made SuiteApps for many countries. But that is not always enough to satisfy every business, so many partners have created additional custom solutions, which fill in the gaps for more demanding client requirements.

For instance, in the US, NetSuite offers a good set of features for tracking tax rates by state but does not provide all of the info clients need to make sure they are charging the right sales tax rate for all of their customers in every state, district, city, and so on. Partners, such as Avalara, have created SuiteApps that manage the calculation of the correct tax rates for sales and other related transactions, so the business does not have to. Check out all of the NetSuite-provided and partner solutions for taxes at SuiteApp.com and talk to the NetSuite sales team for more information.

How you go about setting up and using the native tax features depends on whether the **SuiteTax** feature is enabled in an account. With SuiteTax enabled, companies get more advanced tax features (nexus and rate features, for instance, which are explained shortly), and NetSuite provides starter records for things such as tax types and codes for many countries. Without SuiteTax, you get to set all that up on your own – and keep it up to date over time. Before the feature can be enabled in an account, someone from the business will need to contact NetSuite sales or support to let them know that SuiteTax is needed (or it might be set up during the sales cycle instead). This is so NetSuite can evaluate the business to make sure the feature will be compatible with the needs of the business.

In either case, once you have enabled the features you will use, you can go about the initial tax configuration. Visit **Setup | Company | Set Up Taxes** to establish a few global preferences. Then, set up one or more nexuses. A **nexus** is a tax jurisdiction, or a place where the business must meet a specific set of tax regulations. For instance, if the business of your client has its home in Delaware in the US, then you could set up a nexus for the US, in Delaware. You can set up as many nexuses as are needed to work with the entire business's structure and locations. Next, we set up **tax types** (VAT, state, city, county, etc.) and **tax codes**. NetSuite will help with US tax codes per state, for each nexus you set up earlier. Beyond that, look for a SuiteApp to help with other countries or other special cases.

Here's what the screen looks like when you're setting up a new tax nexus:

Figure 13.7 – The new tax nexus screen in NetSuite – three simple fields

There are quite a few steps to setting up taxes, and since most for-profit businesses are worried about collecting and paying them, it is critical that we get the tax features set up correctly for every new client. Check out NetSuite Help for more information on these topics.

Summary

There are so many accounting- and finance-related features in NetSuite that we could not possibly cover them all in this book. We learned about the GL, currencies, banking, and journal entries in this chapter. You will almost certainly learn more while helping your clients with their implementation, but hopefully, this chapter has gotten you started and given you some pointers in the right direction. Don't forget to explore new features as they're made available, such as the Fixed Assets and Account Reconciliation modules. To become more familiar with these features, read the Help pages and **SuiteAnswers** under the *Accounting* or *Taxation* headings and stay up to date with new features as they are rolled out twice a year. And as always, there are lots of resources outside of NetSuite now as well.

In the next chapter, we will focus on **procure-to-pay** transactions, such as purchases and receipts, vendor bills, and payments.

Self-assessment

Think about these questions in terms of your client work and consider how you would handle these requests on your own:

1. You are beginning to work with a new client when they ask you whether they will need separate NetSuite accounts for each of their legal entities. They are somehow convinced this will be necessary, based on a conversation they had at a convention with another company's CEO. How will you guide them to the right solution?
2. Your client has asked you to set up their account so that currency exchange rates are only evaluated and updated once per year. Is that an option within NetSuite, and if so, how would you help them set it up? (Hint: You would not use the native exchange rate-syncing feature.)
3. You are helping *The Toy Company* implement NetSuite, and they have asked you how they can control the GL impact for their item fulfillments and invoices so that one type of income is applied in a way that NetSuite does not support out of the box. What type of customization might help them achieve what they need?
4. When a company uses a bank that is not in NetSuite's supported financial institution list, we have to set up custom bank file formats for them. Do you know where to go in the **Help** pages to learn more about this?
5. The company you are working with tells you that they must lock down their accounting periods regularly as the accuracy of their financial reports is key. But they also insist that two users need to be able to change any data on transactions, even after its period has been fully closed, going back any amount of time. Can you help them achieve both of these requirements somehow, and if so, how would that affect their financial reports for a period where amounts have been modified?

Procure-to-Pay Transactions

Once you have vendors and items in the account, you can start helping your client learn how to use **Procure-To-Pay (PTP)** transactions: Purchases, Vendor Bills, Payments, Returns, and Credits. For some companies, PTP is only for internal business purchases, but for many businesses, the entire business depends on getting this right, since if they pay too much or do not buy the right products, their sales will be negatively affected.

In this chapter, we will cover the following topics:

- Using requisitions, purchase requests, and purchase orders
- Managing special orders and drop-ship orders
- Utilizing inbound shipments and processing item receipts
- Planning for and using vendor bills
- Tracking vendor prepayments and making vendor payments
- Processing vendor returns and vendor credits

The business buyers, **accounts payable (A/P)** managers, and employees will be your primary contacts as you work your way through these activities. They should have already completed the NetSuite Foundations training before you begin training them further on the A/P features of the system.

Using requisitions, purchase requests, and purchase orders

Every business needs to purchase various things to keep its doors open. A software company might only need to pay for things such as rent, utilities, and office supplies, but a manufacturer or wholesale distributor also needs to purchase at least some of the products they will sell. In *Chapter 11*, we talked about the various types of items NetSuite supports and their various uses. We can enter purchases into the system for any items that were set up as an inventory type item, a non-inventory type item, or service type item, using either **For Purchase** or **For Resale** transactions. Let us start looking at these PTP transactions by covering the various ways NetSuite lets us enter and track purchases.

As we talk about the PTP process in this chapter, remember that NetSuite does not have any screens or features using that phrase. Instead, the system refers to all of the things we will discuss in this chapter as **supply chain management**. You can find that as a top-level heading in the Help system, and you can refer to those pages if you need any additional assistance with these topics.

The normal, minimal process flow for purchases looks something like this:

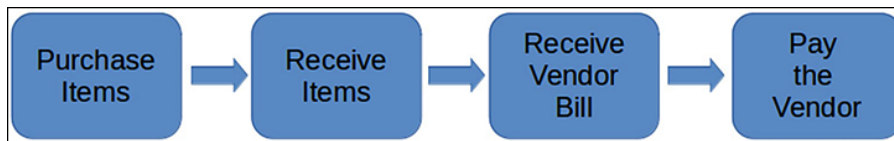


Figure 14.1 – A typical purchasing process for many companies

However, this is not the only way NetSuite knows to handle purchases. There are other steps in the process for some businesses, including **Requisitions** and **Purchase Requests**. We typically see these used in companies where the employees need to request things that they need to do their jobs.

We use requisitions as a sort of pre-purchase transaction. They allow one or many employees to indicate which things they need. They do not have to know which vendor the items are sourced from to create a requisition; they just indicate which items they need and their quantities. Either the employee or someone in the A/P department will specify the amount the company should expect to pay for each item, and then these transactions are typically run through an approval process of some sort. When they are approved, they can be converted into **Purchase Orders (POs)** one at a time or in bulk (or they can be rejected, of course).

Separate from requisitions, we have purchase requests as a method for tracking employee requests for purchases. Employees usually must log in to the *Employee Center* to create a new purchase request, and each transaction is tagged to a specific vendor. Once the employee creates the transaction, it should usually go through the company’s approval process. The native approval process is very simple and not very flexible (it relies on the employee’s supervisor or someone else who has designated the role to approve the request), so this is something we routinely customize, based on the client’s requirements. As soon as a purchase request is approved, it appears in the same place as all POs. Managers can still review them from there, and the employee can always review their status via the Employee Center.

Once a requisition or a purchase request has been converted into a PO, we manage it via the **Transactions | Purchases | Enter Purchase Orders | List** screen. Users with access can also create a new purchase any time they like, directly, of course. Getting PTP users used to working on the PO screen is usually not a problem, especially if you have previously helped them set up the forms they will use. Depending on the client, you might also need to provide a CSV import template file to streamline the bulk PO creation process for them.

For services and software companies, we typically see some POs being used to track non-tangibles such as contractor or consulting expenses. This can easily be enabled on the **Accounting Preferences** screen, and then you will see both *expenses* and *items* sublists available on the PO screen, like this:

The screenshot displays the NetSuite interface for creating a purchase order. At the top, there is a navigation bar with tabs: Items, Billing, Relationships, Communication, Custom, and R&H Commissary. Below this, an 'Exchange rate' field is set to 1.00. The main section is divided into two tabs: 'Expenses 0.00' (selected) and 'Items 0.00'. Under the 'Expenses' tab, there is a 'Clear All Lines' button. Below that is a table with columns: Category, Account, Amount, Memo, Department, Class, and Location. The 'Category' column has a dropdown menu. Below the table, there are buttons: Add, Cancel, Copy Previous, Insert, and Remove. At the bottom, there are buttons: Save, Cancel, Auto Fill, and Actions.

Figure 14.2 – The expenses and items lists enabled on a purchase order in NetSuite

With this configuration, the company can enter purchases as they need, including things such as utilities, bank service charges, time from a contractor, or other services. Expense lines on transactions in the system are always tagged to a GL account instead of an item, and they can also be tied to a project if the account has been set up for project tracking.

For example, if your client is a maker of tax software, they might hire professional tax accountants to assist customers with audits. Those projects would typically be associated with any POs that were created for the consultants they hire.

Inventory and assembly type items that are entered in a PO are generally assumed to be added to the company's inventory when they are received (unless they are drop-ship items, as explained in the next section), and they can have an **expected receipt date**. This is generally very useful since everyone wants to know when to expect to receive the things they order, but every company must decide how this date will be set, and some companies need to take this even further. We sometimes need to break this down with estimates such as when the item will be manufactured and shipped and when it will arrive in the destination country as well. This is more common in certain industries, such as furniture makers/distributors. We can add custom fields in these cases to store the additional data, but then customization (a workflow or script) will be needed if the dates must be auto-populated, either via bulk updates of some sort or from other sources outside the system.

One last PTP transaction worth mentioning is the **Purchase Contract**. In some industries, where parts are purchased frequently and in bulk, it can save the company a lot of money to establish a contract to buy some items on a contract basis, typically at a preset price. Many suppliers would rather know they're going to sell the business a set of items each month, over a span of time, versus dealing with one-off POs. NetSuite supports purchase contracts, and they are used in conjunction with POs. First, we define the contract, setting the expected price for each item included, and then we create POs to record specific purchases at those prices. A purchase contract can incorporate things like tiered discounts, which makes their use very attractive for companies purchasing these items in bulk.

There are a few other features worth exploring around POs as well, including blanket POs. We use these to decide, along with a vendor, whether we wish to buy sets of items over time, and this helps streamline the data entry in the system. Check out NetSuite's Help pages for more information on these sometimes very handy features.

For some companies, we might need to directly relate sales to their purchases. Let us take a look at the two special types of POs we might create for these reasons.

Managing special orders and drop-ship orders

POs are created for all types of purchases, including those NetSuite calls Special Orders. These are orders for items we do not normally stock or treat like another inventory.

For instance, if your client is a plumbing supply company, they might create some items in their account when a customer specifically requests them, and in those cases, they want the system to track the purchase as a special or very rare occurrence. We define these special use cases on the item record versus on the PO. When we purchase these items, they sometimes require some special customization or personalization, such as when a team jersey needs the team's name added to the back. Special orders are typically begun in the system as a sales order (SO). The item is added to the order and when NetSuite knows which vendor is preferred, it will automatically create a PO for the item with that Vendor. The PO is then linked to the SO and the fulfillment is handled when the client gets the special item in stock.

Alternatively, when the manufacturer of an item ships an item directly from their facility to the end customer, we call that a **Drop-Ship** item/order. The process begins with a sales order again, but in this case, we do not directly fulfill the order ourselves. Instead, we just record the fulfillment in the account when the manufacturer notifies us they have shipped the order and then proceed with billing the order from there.

In both cases, tracking additional transactions creates a little more complexity for the users, but generally, they like how NetSuite affords them visibility of each step in the process. These same transactions (Special and Drop Ship Sales/Purchase Orders) can be used for intercompany sales as well, where *Subsidiary A* buys an item from *Subsidiary B*, for instance.

Once items can be purchased, we typically need to then track them and eventually receive them in the inventory. We will look at those transactions next.

Utilizing inbound shipments and processing item receipts

Once a purchase has been created in the system, the business will find a way to communicate that to their vendor, via email, a portal created especially for that purpose, or, in the worst case, via printed paper copies when necessary. The vendors then queue up the work and let us know when we can expect to receive the items. For products coming from another country, as is commonly the case for purchases made from Asian manufacturers and suppliers, we sometimes want to track those shipments as they are in progress. NetSuite offers us **Inbound Shipment** transactions for this. The fields for these records natively allow us to track a status (*To Be Shipped*, *In Transit*, and others), a list of items in a shipment (most commonly, this is a container being transported via a ship), the related *Landed Costs*, and more.

For companies that need this feature, we usually want to associate various sets of items from multiple POs with one inbound shipment and then track its status as the products are on their way to the company's warehouses. One key aspect of this is tracking when the products become our client's property. This can vary, depending on a few factors, but NetSuite offers a **Take Ownership** button in the UI for this purpose. A user is expected to select that button when they know the shipment has reached whichever point in its journey where the company owns the items. This can be automated as well, for businesses that do a lot of international shipping.

Once the items have reached our client's warehouse, each inbound shipment can be received in bulk, and the system takes care of updating the related PO accordingly. Once the items are in stock, we can create the set of needed Vendor Bills in bulk as well. Again, since it is very common to have more than one PO related to an inbound shipment, we will end up with a set of bills created in the end as well.

Whether we are using inbound shipments or not, the main transaction we use to convert items into inventory is the **Item Receipt**. A receipt is a mechanism we use to tell NetSuite that items we purchased are now in our possession, typically in our warehouse or a retail store. Items can be received in a bulk process (via **Transactions | Purchases | Receive Orders**) or we can receive individual orders when needed. The default **Receive** screen is relatively simple to work with, having just a few fields in the header section at the top and then a sublist for items. Note that there is no **Status** field on a receipt; the receipt is only used once we have the items.

This is what the default bulk **Receive Orders** screen looks like:

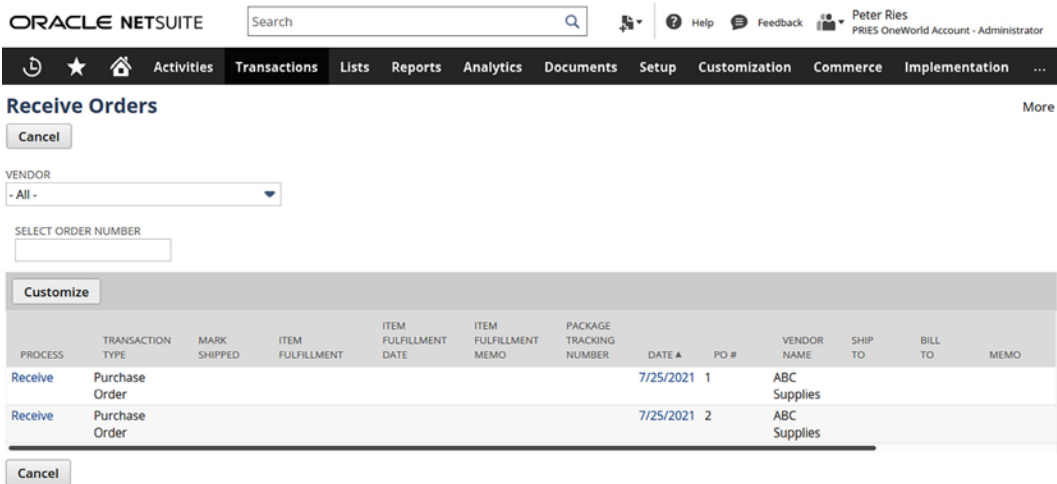


Figure 14.3 – The Receive Orders screen in NetSuite

This screen allows you to select each purchase order or transfer order that is ready to be received so that you can process them one at a time. If you enable the **Advanced Receiving** options, then you can receive multiple orders at once. Whether you have 1 or 100 orders waiting to be received, you can select those that are arriving today, and the system will create the receipts for you. This is very handy if the business usually receives the same quantity they ordered, but otherwise, it can be more trouble than just receiving each order individually.

As you create each receipt, the three most important data points the system is looking for are the *location* where the inventory is now stored, the *quantity* of the items received, and the *cost* you want to associate with that quantity. This cost feeds into the system's cost accounting reporting, so it is typically something we want to focus on at this moment, to make sure we are getting it right. Many companies with inventory consider this a critical part of the business' success, for instance. NetSuite allows us to track item costs in a few ways, but most of those options rely on this amount being entered onto receipts. The costing fields and options are all on the **Item** screen, so we should set up each item for its intended use before receiving anything. Read the Help page on this topic if you are not familiar with NetSuite's supported choices.

An item receipt can also be used to indicate when expenses are incurred, for your utility bills or services purchased, for instance, if the PO has expense items on it.

One other thing to note here is that POs and item receipts are frequently integrated with NetSuite or third-party **warehouse management systems (WMSs)** for larger companies that manage inventory, which makes sense since the WMS needs to know what you have purchased. They are then used to let NetSuite know what was received. (We will talk about fulfillment in *Chapter 15*.) This typically comes in the form of a SuiteApp we install, but this can sometimes require customization to complete as a client wishes.

All receipts can be billed as soon as they are created. NetSuite also offers the option to bill a PO before it is received. In this case, the bill is created before we know the quantity we will receive, so, as you will learn in the next section, matching up the items and quantities on POs, receipts, and bills is very important to NetSuite's manufacturing and wholesale distributor clients.

Planning for and using vendor bills

Vendor bills in NetSuite record the amounts a business owes vendors for purchases. Since vendor bills are part of A/P, that team usually has a lot to say about how they are set up and used. We bring vendor bills into an account via CSV imports and integrations with EDI partners, and can create them manually, as needed. When created directly in the system, we can do this one at a time or use the **Bill Purchase Orders** screen to create them in bulk.

Once they are in the account, the A/P team determines when they need to be paid (some come with terms, some are due on receipt) and batch processes them via the **Payments** interface (as described in the next section).

Some clients do not always need to enter bills for every payable item, though. NetSuite allows us to simply print a **check** for cases where the payment was made in exchange for the goods at the time they were received. If a bakery employee had to go and buy 10 bags of flour from a grocery store, in an emergency, and they paid for the items at that time, there is not necessarily a need for a vendor bill unless the A/P team has decided this is their policy.

Vendor bills also have an optional approval process, just like the other PTP transactions. When vendor bills approvals are needed, we usually see a chain of managers providing the approvals, depending on the bill's total. The approval workflow that comes with the system normally covers most companies' needs, but customizations are possible too. The approval usually includes a review of the PO and item receipt details to make sure they are aligned or that any variances were properly addressed. That is where the 3 Way Match feature comes in.

Most companies I have worked with within the wholesale distribution industry wish to use NetSuite's **3 Way Match Vendor Bill Approval Workflow** feature to help them validate the items, quantities, and amounts on POs and Item Receipts with Vendor Bills, so that they will always know when there are discrepancies. For example, if the PO said we ordered 20 units of an item, and we enter the bill for 20, then we need to know if we did not receive 20 units of that item. In that case, some sort of adjustment is needed to reflect the reality of the receipt. In many companies, the volume of purchase transactions can be high enough that they need a workflow to automate this validation, saving them processing time and increasing the accuracy of their billing and payments to vendors. The vendor might still send us the missing quantity, but at the time the bill is processed, we should only pay for the quantity we received – usually. Check out the section called *3 Way Match Vendor Bill Approval Workflow* in the NetSuite Help pages for more information.

Paying for what we purchase is a necessary part of the PTP process. The system does what it can to streamline this part of the process and make it as simple as possible, but there are a few things to watch out for, as we will learn in the next section.

New feature highlight: Automatic Bill Capture

With the last NetSuite release, businesses can now utilize **NetSuite Bill Capture** SuiteApp to automate the importing and recognition of bills from their vendors and suppliers. Users drag a vendor's invoice email into a NetSuite screen, and the system reads the text, recognizes the important bits, and takes care of creating a vendor bill for them. This uses the most recent AI text recognition features offered by Oracle's cloud services and the service learns to read the client's bills better and better over time. These bills are even matched to open POs and run through the approval process automatically! Bill Capture is available for an extra fee but is worth looking at for clients who receive many bills from a large number of vendors.

Tracking vendor prepayments and making vendor payments

Every company with vendor bills must pay them, somehow, at some time. But some businesses also need to prepay for their orders from some of their vendors.

These can be called deposits or prepayments by the client, but in the system, we refer to them as **Vendor Prepayments (VPPs)**. They are entered like other payments, except that each VPP can be applied to just one PO. NetSuite can automatically apply these VPPs to the vendor bill when it is created if you have enabled that option (in **Accounting Preferences**). We can disable this feature if the client prefers to apply VPPs to POs manually. NetSuite tracks the list of VPPs for each vendor and makes it easy enough to associate them with Vendor Bills as they are created, but generally, it is easier to let the system apply the VPP to the bill for you.

When the time arrives to train users on creating **Vendor Payments**, we show A/P users how to use the **Pay Bills** and **Bill Payments** screens to create vendor payments in bulk whenever possible. This allows them to create the payments for a set of bills all at once while applying filters to the list of open bills. This is what that screen looks like, except most clients will usually have a long list of open bills they could be creating payments for:

DATE DUE	TYPE	ID	VENDOR	REF NO.	CURRENCY	EXCHANGE RATE	ORIGINAL AMOUNT	AMOUNT DUE	DISC. DATE	DISC. AVAIL.	DISC. TAKEN	PAYMENT
7/26/2021	Bill		ABC Supplies		USA	1.00	1,974.00	1,974.00				1,974.00

Figure 14.4 – The Bill Payments screen for applying payments to open Vendor Bills

On this screen, A/P users can choose which bills to pay, and they can enter the amount of the payment and enter a discount if any discount can be applied.

There are a few things to note about applying payments to vendor bills. Let us take a look:

- **Posting periods:** Like any transaction that will have an impact on a GL, **Bill Payments** has a **POSTING PERIOD** field, and this will always default to the current period. Users can select a different period when they need to.
- **To be printed:** If you need to print a check for any payment, you can select the **TO BE PRINTED** checkbox and the system will queue up the checks for a user to print in bulk from a separate screen (**Transactions | Management | Print Checks and Forms**).
- **GL impact:** After each payment has been saved into NetSuite, you will be able to view the GL impact (debits and credits per account) on a subtab while viewing the payment.

When we use the **Pay Bills** screen to create a set of payments all at once, we enter all the payment details, select the bills to be paid, and then submit the form. NetSuite processes those in a batch, in the background. We can check on the status of the *Bill Payments* batch, or any such bulk operation, by selecting the **Transactions | Payables | Pay Bills | Status** menu. That screen tells us how many payments have been created and, if applicable, if any errors occurred during that processing.

One other handy feature related to bill payments is **Electronic Bank Payments**. This NetSuite-offered SuiteApp enables clients to pay bills electronically through the bank they've set up in their account, and this replaced the EFT feature previously available in the system. Streamlining the process, from creating the bill to paying the vendor for it, organizations can save a lot of time and trouble by automating this process but sticking to their approval process and making sure only the right roles are allowed to perform electronic banking functions. This feature usually requires some technical help to set it up, but once everything is configured and tested with the client's bank, automating payments to vendors can really help A/P departments with their work. If the NetSuite offering doesn't work for your clients, there are quite a few other alternatives (that also enable **Electronic Funds Transfer (EFT)**) available in the SuiteApp marketplace. Just search there for *SuitePayments* to review the available options. Once the A/P users are familiar with processing bills and payments, it is time to make sure they can also process returns and credits when they need to.

New feature highlight: Transaction Line Distribution

New in the last NetSuite release, we now also have access to this new SuiteApp. Once installed, the app allows users to spread costs on a vendor bill across multiple subsidiaries or other segments. Fields are added to the **Bill** screen, allowing users to quickly assign expenses to one or more business units, based on pre-defined rules or on the fly when needed. For instance, say your client receives one bill for rent, but multiple subsidiaries share that office space. With this SuiteApp, one bill record can assign that cost to each of the subsidiaries, in appropriate proportions, based on a template, if you like. Natively, a bill in NetSuite can only be assigned to one subsidiary still, but the app generates journals to track the split-up cost in an hourly batch process behind the scenes.

Processing vendor returns and vendor credits

Vendor Return Authorizations (or just **Returns**) are transactions we use in NetSuite to tell the system when an item we purchased needs to be sent back to the supplier. We can create them either directly from a PO screen (once the PO's items have been received) or by selecting **Transactions | Purchases | Enter Vendor Return Authorizations**. We typically use these when an item has been received in a damaged state, or sometimes a company has an agreement with its supplier that allows it to return some number of unsold products. Let us take a look at this process:

Once a vendor return is created in the system, the items need to be *fulfilled*, which means shipped out to the vendor. This can be done via the **Return** button on the **Vendor Return** screen. Clicking that takes us to a new **Item Fulfillment transaction** screen. We enter the details of exactly what was shipped out and then the return is complete.

Once a vendor return has been fulfilled, it moves to the **Pending Credit** status. This means we can now create a **Vendor Credit** transaction for the items on that return. We can always create standalone vendor credits too, for situations when we want to record a credit without performing a return first. Vendor credits, or as NetSuite sometimes calls them, **Bill Credits**, are used to reduce the payable account balance for the vendor, which you might need to do for a variety of reasons.

For example, if you have \$1,000 of open vendor bills from one vendor, but you negotiated a discount with them over the phone for some of those orders, you might enter a credit to reflect that discount in NetSuite.

On a vendor credit, we can then indicate the expenses or items we are getting the credit for, or we can use the **Apply** sublist to select any open bills we are receiving this credit for. We do not have to tie a credit to a bill, though, since there are times when the credit will not be associated with any set of bills. NetSuite will generally try to pre-enter the expenses, items, or bills to apply a credit to for you, depending on how you initially create it.

In this section, we talked about how to process returns and credits for vendor purchases. That wraps up our discussion of the PTP transactions and processes within NetSuite. There is a lot more you can learn, as with all the topics in this book. I suggest that you explore NetSuite Help and the training videos they provide if you want to explore any of these features in greater depth.

Summary

The **Procure-to-Pay** process within NetSuite is usually robust enough to meet the requirements of both large and small NetSuite clients. It includes a set of transactions that allow us to track and manage vendor purchases, receipts, payments, and returns. There are always going to be places that some businesses need to customize, but the native features provided here are very strong as a starting point. I hope you have learned how to talk to your clients through the PTP screens in the system since, without a solid foundation for buyers and A/P users, the business cannot be successful on the sales or A/R side.

In the next chapter, we will move on to the **Order-to-Cash** process and its transactions: Estimates, Sales Orders, Fulfillments, and more.

Self-assessment

When you work with a variety of clients, you know they will each have unique requirements, and sometimes, they can present some real challenges. Answer these self-assessment questions as if you are the one who must solve the problem for a client. Be creative and have fun answering these questions:

1. Your client provides services to legal practices. Occasionally, they must hire an outside contractor to provide the expertise they lack. Which of the purchase transactions described in this chapter might suit this purpose?
2. *ABC Co.* sells electronics to the transportation industry. One of their customers asks them to provide a slightly modified version of a GPS tracker. Which transaction might be the right way to track their purchase of these devices?
3. When your client buys widgets from one of their suppliers, they must place a 50% deposit at the time of the PO. They typically place multiple orders for those parts. Can they create one Vendor Pre-Payment for all the POs they enter at any one time?
4. Your client is using Withholding Taxes on Vendor Bills, and they occasionally see a message saying *Your transaction does not balance* on that screen. What should they do to correct this? (Hint: Search in SuiteAnswers for some of these keywords if you are not familiar with this.)
5. *DEF Inc.* buys a lot of parts from one main vendor, and they frequently return parts to them as well. They process those returns outside of the system, but they still need to record them in NetSuite. How can you help them streamline the vendor return and credit processes so that their users can make these entries as easily as possible?
6. Your client is looking for help processing the hundreds of one-line bills they receive from their vendors each month. How can you help them get those bills into NetSuite, and then how can you help them pay them in bulk processes to avoid a lot of manual work?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search for this book by name.

Note: Keep your purchase invoice ready before you start.



Order-to-Cash Transactions

Every company has something they think of as their sales (which makes up their **accounts receivables (A/R)**), even if they are a non-profit business. For this part of the business, we have the **order-to-cash (OTC)** transactions, including Opportunities, Estimates, Cash Sales and Sales Orders, Fulfillments, Invoices and Payments, and Returns and Credits. We tend to spend a good percentage of time on any implementation project working out the details of these transactions, so get ready for a deep dive into all things order-related in this chapter.

In this chapter, we will cover the following topics:

- Using opportunities and estimates before sales
- Recording sales via sales orders and cash sales
- Managing item fulfillments for sales, transfers, and returns
- Making the most of invoices
- Entering customer deposits and payments
- How to track return, credit, and refund transactions

Your primary contacts for this work will be the client's sales and customer service people and managers. They should have already completed the NetSuite Foundations training before you begin training them further on the system's A/R features.

Using opportunities and estimates before sales

For some businesses, sales to new and existing customers are not a sure thing until they have negotiated quite a bit and worked out all of the details. If this applies to your client, you will want to help them set up and learn how to use **Opportunities** and **Estimates** in their accounts.

Opportunities are used to track potential deals; the details typically start fuzzy and tend to get easier to set over time. The items, quantities, and prices can change as negotiations take place.

Here is an example of the top set of fields on a new opportunity:

The screenshot displays the NetSuite 'Opportunity' form. At the top, there's a header with the 'Opportunity' title, a search icon, and buttons for 'Save', 'Cancel', and 'Actions'. Below this is the 'Primary Information' section, which contains several fields: 'OPPORTUNITY #' (labeled 'To Be Generated'), 'TITLE', 'COMPANY' (with a star icon), 'JOB', 'SALES REP' (dropdown), 'PARTNER' (dropdown), 'STATUS' (dropdown with a star icon, currently showing a list of options: Proposal, In Discussion, Identified Decision Makers, Proposal, In Negotiation, Purchasing, Closed Won, and Renewal), 'PROBABILITY' (star icon, set to 50.0%), 'EXPECTED CLOSE' (star icon, set to 8/30/2021), and 'WIN/LOSS REASON' (dropdown). A 'DETAILS' section is also visible. Below the Primary Information is the 'Forecasting' section, which includes 'PROJECTED TOTAL' (star icon, set to 0.00), 'RANGE' (set to 0.00), 'TO' (set to 0.00), 'FORECAST TYPE' (dropdown, set to Omitted), and 'WEIGHTED TOTAL' (set to 0.00).

Figure 15.1 – A new opportunity and its STATUS list in NetSuite

Opportunities can be created for deals that might never happen, so they are generally regarded as uncertain until they reach the *closed/won* status. These transactions also have a field that is used to track how confident we are about closing them, called **PROBABILITY**, which is expressed as a percentage. **STATUS** and **PROBABILITY**, when used together, allow the salesperson tracking them to quickly communicate how close we are to making a deal. This is key for sales reporting in businesses that use these transactions, since commissions are common for salespeople who close opportunities.

NetSuite also includes **Forecasting** fields on the **Opportunity** screen, allowing for a range of values (the transaction's total) to be guessed at and allowing us to choose whether to include the value in the salesperson's forecasts. These lists and fields can be customized to use whatever ranges and terms the client's sales team is most used to. One other important thing to note about opportunities is that they do not have shipping fields by default. Natively, NetSuite does not include any shipping charges on these transactions, since they are not meant to track specific sales but a general potential for sales down the line.

Estimates (which are sometimes called **Quotes**) are used when we need to make an offer to a customer for a specific set of items and see whether it is accepted. These transactions are typically used in government purchases for physical items, or any industry where larger, more expensive sales are made. The default **Estimate** form has the same **STATUS** and **PROBABILITY** fields as opportunities do, as well as having **Shipping**, **Billing**, and **Accounting** subtabs. It does not have the **Sales** subtab that opportunities do, since fields such as **Sales Readiness** and **Buying Time Frame** make no sense here. The item lines add fields for tax calculations.

One other use I have seen for opportunities and estimates is custom pricing. In these cases, the client wants to negotiate special per-item pricing with some of their customers, and they use opportunities and/or estimates for this purpose. They add custom fields to the item lines to track which items/prices were *won* or accepted and approved for use on other sales transactions down the line. Users can then find the source transaction, select **Create Sales Order** (for instance) from there, and then all of the prices from the lines are copied onto the new transaction. If the client wants to have the option of building all new sales transactions, using any previously approved items/prices from estimates, then that requires customization.

As you have seen, these transactions are very handy to track sales before they are officially completed, but once the customer is sure about what they want to buy, we typically record a sales order, so let us take a close look at that next.

Recording sales via sales orders and cash sales


NetSuite offers two transactions to record completed sales: **sales orders** (sometimes referred to as **SOs**), which are used when the items in an order are not delivered to the customer at the time of the sale, and **cash sales**, for when they are delivered and the payment is received at the same time. Clients in the wholesale distribution, manufacturing, software, and services verticals usually use SOs. Retail, e-commerce, and **point of sale (POS)** clients frequently use cash sales and invoices to record their sales.


SOs in NetSuite tend to be treated like a Swiss Army knife, in that they can serve so many different purposes, depending on how the business needs to use them. Training users on these transactions starts with the basics, and that starts with how they are created. An SO can be created from an opportunity or an estimate via an import, or via an import from some other system (such as via web services). The clients I have worked with over the years have used all of these methods, so it is good to remember how many options there are – and how common it is to use more than one within any single business.

Here is the default set of fields on a sales order item line for reference:

ITEM *	QUANTITY	UNITS	SERIAL/LOT NUMBERS	DESCRIPTION	PRICE LEVEL	RATE	AMOUNT	COMMIT	COMMITMENT CONFIRMED	ORDER PRIORITY	TAX	OPTIONS
Badminton Set	4				Custom	27.99	111.96	Available Qty				
101	1			Sports set, Misc	Base Price	101.00	101.00	Available Qty				
<div><div></div></div>			<Type & tab for single value>									
<div><div>Add</div><div>Cancel</div><div>Copy Previous</div><div>Insert</div><div>Remove</div></div>												

Figure 15.2 – The fields we can see on the item line on a sales order by default

 **Quick tip:** Need to see a high-resolution version of this image? Open this book in the next-gen Packt Reader or view it in the PDF/ePub copy.

 **The next-gen Packt Reader** and a **free PDF/ePub copy** of this book are included with your purchase. Scan the QR code OR go to packtpub.com/unlock, then use the search bar to find this book by name. Double-check the edition shown to make sure you get the right one.



Sales orders tend to be one of the most heavily customized forms for most NetSuite clients, with many custom fields, lists, and sometimes subtabs added per the requirements. This is both because **orders** are the central place to track things such as prices, items, discounts, taxes, rebates, and more, and because they indirectly affect other things such as inventory levels, billing schedules, and revenue.

The following subsections cover a few examples of the many ways that SOs are tweaked for various purposes by different types of NetSuite clients.

Bulk/blanket orders and order guides

Some businesses allow their customers to order items in **bulk** or on a **blanket type** of order and then have them shipped out in smaller sets over time. A native SO can serve this purpose, with just the addition of a few custom fields for tracking the dates and delivery timeline, for instance. NetSuite allows us to partially fill each line of an order over time so that we only ship what the customer is ready to receive every week. **Order Guides** is a NetSuite-provided SuiteApp that makes it easier for customer service representatives to place new orders for items the customer bought previously.

Drop-Ship Orders/Special Orders/Special Work Orders

In *Chapter 14*, we talked about how these special types of orders have an effect on a business' purchasing, and now we can cover the OTC impacts too. NetSuite allows us to tag each item as available for drop shipments and special orders, and if it is an assembly-type item, it can be tagged for special work orders to be manufactured in-house as well. If an item has a vendor selected as the preferred option on the item record, then the system will automatically create a Drop Ship Order or Special Order (which is a special **Purchase Order (PO)**) as soon as an SO is created and reaches the approved status. The SO and the PO are linked in this case, so we can track the fulfillment of the items, even though someone else will ship them to the customer (for drop-ship orders).

If your client has an item that requires personalization or customization before it is shipped out, they will most likely want those items to be tracked as special orders in the account. This can apply to normal inventory-type items and to assembly items as well, in which case the system will create a **Work Order** for the items in a new SO.

E-commerce sales orders

Orders placed via the customer's web store come into NetSuite as SOs, and from there, they are treated like most other SOs. If the client has purchased NetSuite's SuiteCommerce or SuiteCommerce Advanced store, then this happens automatically as soon as the checkout process is completed in that interface. Otherwise, orders are typically integrated into the system from an external web store via some connector or SuiteApp. This is one place where we have to be extra careful with the custom fields and custom actions (scripts) we associate with SOs. We usually want to include only some of those in the web store shopper's experience. E-commerce performance is directly tied to completed sales, and if we make the web store too complicated, we can negatively affect the client's sales.

Project/service deliverables

Services businesses typically use SOs to record the initial sale of the items and services included in a project. These SOs then show what the deliverables for the service work should include. They are used in conjunction with the project record to track things such as the resources that have been assigned to the project, the billing schedule, and, frequently, the revenue recognition plan (see the *Revenue recognition* section for more on that). NetSuite will link these SOs to the project but will not automatically update the project when/if the items on the orders change over time. Since project work is also commonly tied to a contract of some sort, using an add-on module such as NetSuite's SuiteBilling can help you manage the complexities of the set of transactions involved. This is commonly needed by service companies and those who deal in subscription-based offerings.

Promotions

Many clients today need to offer a variety of discounts and promotions to garner as many sales as possible from their customers. NetSuite offers a handy SuiteApp for this, known as **SuitePromotions**, but many businesses have different requirements, so third-party SuiteApps and customer-specific customizations are common in this space as well. We can easily add discounts to the transaction and at the item line level with native NetSuite, and then we can customize the UI and the process around tracking these promotions from there, depending on the needs of the client.

Revenue recognition

Software and services operations require special handling for the revenue on their SOs. Even though these are not seen as *posting* transactions (they do not have a direct **General Ledger (GL)** impact), they still affect how revenue is recorded in the system, and these types of businesses need to follow strict guidelines around this. If they are a publicly traded company, then this is even more true, and mistakes that are made in this process can cost them a lot of money and potentially new business.

Revenue recognition is really an accounting concern, and as such, it touches on more than just SOs in NetSuite, but SOs are typically the beginning of this process for these companies, so the focus starts here. This topic can be relatively simple or become very complex, especially for companies that will typically see their sales and billing performed in separate accounting periods. If this subject is brand new to you, you can read about it online in places such as Wikipedia and on the **Financial Accounting Standards Board (FASB)** or **Generally Accepted Accounting Principles (GAAP)** reference websites (see <https://www.fasb.org>). The laws and regulations that apply to your client will also vary, depending on the part of the world they are in.

NetSuite allows us to enter payments for SOs when that is appropriate, but most of the time, we use a **Cash Sale** transaction instead when we want to record the payment along with the sale. Cash sales are like SOs, except they include an assumption that the goods that were sold were delivered to the customer at the time of the sale and that the payment was received in full. If neither of these things is true, use an SO instead. We see cash sales used, for instance, when a retail store sells a pair of jeans, or when a services company wants to record the delivery of billable items, time, or expenses that have been paid for. Just keep in mind that a cash sale cannot be fulfilled, but it can be returned and processed for a refund if needed. I will cover those topics later in this chapter, in the *How to track Return, Credit, and Refund transactions* section.

New feature – Configure, Price Quote or CPQ

In 2023, NetSuite acquired a product from another company and now offers its own in-house CPQ tool. With this, order entry for highly configurable products, from furniture to skateboards and much more, is now easily achieved in a custom UI used in addition or instead of the native **Sales Order** screen. With the CPQ module, clients can allow inside sales reps or customers on the e-commerce platform to select the products they wish to order, configure them to their exact specifications, and see dynamically applied pricing as they configure their purchases. Imagine a furniture company that sells a line of desks. They want to let their customers select not just the product, but the style, color, wood finish, size, and number of drawers for each item, so they receive the perfect desk to suit their own needs. With CPQ from NetSuite, setting up the system to drive this kind of customization is quick and efficient, and the result is more satisfied customers and more sales! Search NetSuite Help for *NetSuite CPQ* to learn all about how helpful this feature can be for your clients.

Once your client is tracking their sales in NetSuite, they will want to report on them in various ways. The system comes with a nice set of starter reports that most clients will find useful. Just take a look at any of these menu options to see the variety NetSuite offers: **Reports | Order Management, Sales or Orders**.

Going back to SOs, they can, and almost always need to, be fulfilled to make the sale final, so let us explore that topic next.

Managing Item Fulfillments for sales, transfers, and returns

I mentioned earlier that an SO is not a posting transaction, and that is because, as per most modern accounting standards, the sale is not final until the goods or services that were sold are delivered to the buyer.

For physical goods, we record that event in the system via **Item Fulfillments**. These are relatively simple transactions with a copy of the items list from our SO, for those items that are configured as fulfillable and either inventory or non-inventory type items. Sometimes we do want to *fulfill* non-inventory items, such as in the case of software or some other item we sell but don't store in a warehouse. Remembering that SOs are not GL posting transactions, we sometimes use item fulfillments to record the transfer of the items, even when we didn't ship anything.

For clients who do a lot of shipping, you'll want to be familiar with setting up shipping-related things in their NetSuite account. That will include *Shipping Items*, which are the methods by which they ship products, *Shipping Carriers*, and integrations with carriers. All of this is well documented in NetSuite Help, under *Integrations*, the section about the integrations the NetSuite system supports for all clients, and then there are a number of less common third-party shipping integrations you can add to an account with a SuiteApp from the Marketplace.

There are a lot of variations to this process, depending on the company and how they deliver their goods to customers, but generally, sometime after the sale is made, the goods are sent to the customer, drop-shipped, or manufactured and then shipped or similar. We can tell the system which items were delivered, and for warehouse-based companies, we can assign statuses such as **Picked**, **Packed**, and **Shipped** to track each step of the process. When we use these statuses on item fulfillments, we frequently also use **Pick Tickets**, a specially formatted page layout used as a guide by the warehouse pickers. Then, we also use a **Packing Slip** page template when the client wants to include that paper inside the shipment box. Both of these documents have advanced PDF templates you can start from and customize as needed for each client.

Since 2022, NetSuite also offers a mobile shipping solution they call **Ship Central**. Clients who want to tie their fulfillment processes in the warehouse to their ERP with minimal fuss can use this mobile application. It includes features that help with selecting the best rate from multiple shipping carriers, rerating (changing the ship method on an order to get a better shipping price), shipping locally and internationally (once set up), and much more. With Ship Central and optionally the **NetSuite warehouse management system (WMS)**, clients can automate most of their previously manual warehouse processes to get more done with fewer workers. Integrations with shippers are included, so users can print labels as they pack boxes, and track shipments once they're out the door. Search for the NetSuite Help topic *Ship Central Shipping* for more information on this new feature.

Moving on, we typically do not bill customers for orders until individual items on the order have shipped. Most companies that ship products also tend to use a WMS, so we typically see item fulfillments created in NetSuite via integration when a WMS is used.

For more use cases on how we might use fulfillments, check out the NetSuite Help page for topics such as *Automatic Location Assignment* (which is very handy for companies with both retail stores and warehouses), *Fulfillment Requests* (when you want to notify a warehouse in advance of the need for a shipment), and *Multiple Shipping Routes* (for customers who need to record a different shipping method/address on each SO line item).

An item fulfillment can be created from other transactions as well, including transfers and returns. **Inventory Transfers** and **Transfer Orders** are records that are used to track the movement of inventory items from one location (a warehouse, commonly) to another. When the full native process is in use, a transfer order is created first, and then that gets fulfilled via an item fulfillment. An item receipt is created to record when the items arrive at their intended destination. An inventory transfer transaction is the shorthand version of that multi-step process, since it records the move in one step, so an item fulfillment does not need to be created.

When a customer returns an item to the selling company, we typically record the return as a **Return Authorization** first and then record an item receipt to indicate when we received the damaged goods in the warehouse. If the return process includes sending the customer a replacement item, that is recorded on a new SO, and that then gets fulfilled. Companies that allow this sort of thing want to be able to differentiate new orders from replacements, and they include a flag indicating this on the item fulfillment.

Once the sale has been recorded and (when needed) the fulfillment has been completed, we can bill the customer for the sale. We will talk about invoices next and look at all the options we have when using them.

Making the most of invoices

For many companies, creating and sending **invoices** is just as important as making the sale, since these companies did not get paid at the time of the sale, and the invoice is their way of telling the customer how much is now owed. This is very common among companies that allow their customers to make purchases on **terms** (such as Net 30, 60, or 90). It's also possible, though, to create what we call **Standalone Invoices**, those that are not tied to an SO. We might use these to record special items that are billed outside of the normal SO processing. Standalone invoices affect inventory, although they do post to the GL, as with all invoices. This is yet another example where, depending on a business' needs, NetSuite is very flexible while still applying consistent accounting principles to the features it offers. For more info on this, search the NetSuite Help for *Billing Schedules*.

Alternatively, when a billing schedule is in place, we are billing for goods and services delivered over time. The invoice looks just like an SO for the most part; they have the same fields and lists and subtabs. By default, NetSuite requires that an item fulfillment be created for any sales before they can be billed, but there is an accounting preference that allows us to change that rule. This is handy for clients who generate an invoice right after the sale and then deliver the goods separately. For instance, some industries, such as food services, want to print the invoice at the time of fulfillment and include a paper copy in the truck, along with the food being delivered.

Companies in the US (and some other countries), in the software and services industries, may have different requirements for invoices. For starters, they might create multiple sales for their customers throughout a given accounting period. Then, these companies might need to send one consolidated invoice per customer (typically) at the end of that period, though, so we usually set up the offering from NetSuite called **Invoice Groups**. With this tool, we still have NetSuite generate one native invoice per sale (so, many invoices per customer, per month, for example), and then we generate one “virtual” invoice, which is what we send to the customer. This way, they see all of the items they have ordered from us listed on one bill. Payments can easily be applied against the native invoices, but the virtual invoice is updated as well, for tracking and reporting purposes. This is also where billing schedules come in, since customers on a contract need to receive a bill for some of their charges, once per month or on some other frequency.

In the EU and some other countries, though, a virtual invoice is not allowed per e-invoice regulations. Since invoices affect revenue (we typically recognize revenue once items have been billed on an invoice), they play a big part in the day-to-day operations of companies restricted by these rules. I have customized the GL impacts for several clients over the years, based on exactly how they have sold and delivered their goods and services to their customers, variations in the contracts they had, and other factors such as simple formatting preferences on how the invoices had to look when received by customers. For instance, if the business sells services for some special purpose, a custom GL impact may have been made on the sale, deferring the recognition of the amounts until the time of the invoice, when the deferral is reversed.

There are also quite a few companies that want invoices to be created automatically, either when the sale is recorded or when at least some of the items on the order have been fulfilled. In these cases, clients can always use the native **Bill Orders** screen to generate an invoice for every order that is ready for one, but that still requires a user to know how to do that. Otherwise, we frequently create automations, which will generate invoices for the users, according to a business’ rules and timing preferences. The invoice generation step in the OTC process also typically requires quite a lot of advanced PDF template adjusting to make sure that the page layout, formatting, and contents all meet the client’s requirements.

Once the company comes to terms with generating invoices, they need to know how to receive and record payments into the system, so that is the next stop on this OTC journey.

Entering Customer Deposits and Payments

Some businesses want to make their customers pay for part of a sale in advance, and in these cases, we can use the NetSuite native transaction known as **Customer Deposits**.

This is what the default **Customer Deposit** screen looks like:

Customer Deposit 🔍

Save Cancel Actions

Primary Information

CUSTOMER * 6 ABC Customer Co	CURRENCY * USA	<input type="radio"/> UNDEPOSITED FUNDS
SALES ORDER	EXCHANGE RATE * 1.00	<input checked="" type="radio"/> ACCOUNT Checking
DEPOSIT # 2	DATE * 8/29/2021	MEMO
PAYMENT AMOUNT *	POSTING PERIOD Aug 2021	

Classification

SUBSIDIARY Parent Company	CLASS	LOCATION
DEPARTMENT		

Payment Method Relationships Communication

PAYMENT METHOD	NAME ON CARD	AVS STREET MATCH
CHECK #	CARD STREET	AVS ZIP MATCH
PAYMENT PROCESSING PROFILE	CARD ZIP CODE	<input type="checkbox"/> RECURRING PAYMENT
CREDIT CARD SELECT	<input type="checkbox"/> CC APPROVED	
CREDIT CARD #	P/N REF.	
EXPIRES (MM/YYYY)	AUTH. CODE	

Figure 15.3 – Customer Deposit with the default, native fields

These are just like customer payments, except they do not have an **Apply** list and they are usually tied to a single SO. The system will automatically apply a deposit to an invoice once that is generated if it is linked to an SO. Deposits are frequently used for larger, more expensive items, and for service work where a business stands to lose money if the customer changes their mind before the work is completed. We had a client once who sold high-end saunas to consumers and businesses, and each cost more than 5,000 USD. The company required a 50% deposit to be made at the time of the sale, and then the balance was due on delivery.

This is certainly not the only way deposits are used in the system, though. We can use them any time we need, and they can be created directly from an SO in the UI, or as a standalone transaction if we prefer.

Customer Payments are received by businesses in many ways – via checks in the mail, cash tendered at a register, credit card transactions, online payment systems such as PayPal and Apple Pay, and more. No matter how they happen in the real world, we need to enter a customer payment into NetSuite for each payment, to record the items that were being paid for, the amounts, and other details. A customer payment can be applied to any of the open vendor bills on the account, and also to customer deposits, credits, and other transactions, such as journal entries.

We define credit cards on customer records. NetSuite stores the details securely, so don't expect to be able to copy them out of the system, for instance. Most of the time, this is a really good thing, since NetSuite has to be compliant with **PCI** (short for **Payment Card Industry**) standards in order to process transactions through third parties, such as credit card payment services. Occasionally, that security can lead to issues with integrations or customizations, but most of the time, we rely on it, and you should never take any steps that would weaken this security.

Here is what this looks like in the system:

Apply Payment Method Relationships Communication

Payment Amount *

0.00

☐ Auto Apply

Invoices 0.00 • Credits 0.00 Deposits 0.00

Select Item

Date

From

To

All

☐ Pay All

☐ Auto Apply

Apply	Date ↑	Type	Ref No.	Orig. Amt.	Amt. Due	Currency	Disc. Date	Disc. Avail.	Disc. Taken	Payment
<input type="checkbox"/>	4/15/2025	Invoice	114923	1,699.00	1,699.00	US Dollars			<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	4/27/2025	Invoice	885192	123,499.99	123,499.99	US Dollars			<input type="text"/>	<input type="text"/>
<input type="checkbox"/>	4/30/2025	Invoice	115817	1,699.00	1,699.00	US Dollars			<input type="text"/>	<input type="text"/>

Figure 15.4 – The Apply sublist on the Customer Payment screen

For instance, consider this scenario: A customer made five sales in the previous month. They deposited 33% of the total sales. The customer was sent an invoice at the end of the month for the remaining amount due. Now, they are paying the total amount due via EFT.

One payment can be created and applied to a list of open invoices, and the deposit can be applied to that payment record as well. If credit was issued at some point in time for damaged goods, that can be applied at the same time.

Payments can go into the **Undeposited Funds** account or, if the business knows the correct choice at that time, entered into a specific bank account.

Note

NetSuite will only allow us to apply a payment to invoices that use the same A/R account. Hence, I have often been confused when testing a setup by not seeing the Invoices I expect to see on the **Apply** sublist, before remembering this rule!

Assuming all goes well with a sale, this might be the end of the road – the company sold the items, delivered them to the customer, and got paid. But in reality, for many of our clients, there is always the possibility of a return, so let us address that next.

How to track Return, Credit, and Refund transactions

When things go south with sold items, they can usually be returned. A company needs to take the items back into inventory and usually give the customer some form of either *Credit*, to be applied against a future purchase, or a *Refund*. Each business decides on the right option for each of its items, depending on the conditions of the sale, the problem being reported by the customer, and other factors.

The process that is followed in NetSuite in cases like this will vary, but generally follows this flow:

1. If the company is using NetSuite's customer service features, it might record a **case** record during a call or when the customer's email is received.
2. From this case, they can create a return authorization, indicating that they have given the customer the OK to return the product to them.
3. Next, they might create a **credit memo** to show that the customer is due some portion of the money they paid initially against a future order. They might create that new order right away if they need to send out a replacement product. Alternatively, they might issue a **customer refund** if the customer has requested their money back.

This is one way to get through these transactions, but each company tends to tweak this to suit its needs. For instance, we have helped more than a couple of clients create refunds automatically as soon as the items were received with a return authorization. Alternatively, credits can be created as standalone transactions. We might do this while on the phone to a customer, or to issue credit to the customer to correct a mistake on their last Invoice, for instance.

In any case, the system provides us with great flexibility in how we can apply these transactions to each customer, applying just the adjustment that we need to nearly every business case.

Summary

As I have said before, knowing your customer's requirements should always precede any steps you take in NetSuite, and OTC transactions are no different. Using those requirements as your guide, you can always get through this phase of implementation, but getting this right takes practice and determination as you work with each new client. From the frontend with opportunities and estimates, where we are focused on the sales team's needs; to the sales orders and cash sales, where sales, customer service, and accounting all have a say in how things are done; users must become comfortable entering these transactions before they are ready to go live. Adding in the invoices, payments, returns, credits, and refunds usually sums up to a lot of learning in a short time, but it is all worth it when the users confirm that they have become confident in their new skills. By combining your consultancy, the configuration adjustments, and the customizations that your team and others create with the training the users receive, you will achieve this goal every time.

Self-assessment

Here is another opportunity to think about what you learned in this chapter by applying your knowledge to a few hypothetical situations:

1. A software company you are working with wants to enter deals into the system before they are finalized. Their sales always consist of a few software items, services for things such as installation and training, and a maintenance contract for ongoing support after the sale. Which sales transactions will you recommend they use for all of this, and which NetSuite add-ons should be considered to supplement the system's native features?
2. A university you are working with already sells books in their bookstore via their live NetSuite account and SuiteCommerce. Now, they also want to offer digital book rentals via a third-party service. They come to you looking for advice on how to set up the items for these digital books and how to record the rentals in NetSuite. When a student enters an order on the web store for one of these book rentals, the university forwards it to the book rental company's management website. The money is paid to the third party, who then returns a portion to the school as part of their business arrangement. What are a couple of ways you could approach helping them with this?
3. One of your clients wants to offer their clients free shipping on all orders over \$100 and for some specific items, regardless of the amount purchased. NetSuite has features covering this, but the client wants to track their costs concerning their shipping, for reporting reasons. What steps should you take to help them set up the shipping options and then track their shipping costs versus items sold, for their financial reports?

-
4. Near the go-live date for one of your implementations, you learn that the company needs to consolidate its invoices by customer, department, and the customer's PO number. In other words, if a customer of theirs had 10 sales in the last month, some assigned to one department and some to another, all under two different PO numbers, they would want to see a total of four Invoices sent out that month. What natively supported options can you consider that can help them get something into the account quickly to solve this need?
 5. While you are showing a customer how to manage returns, they complain that there are too many steps involved and ask you to do what you can to streamline the process for their users. You start by showing them return authorizations, receipts, credits, and refunds. Think about how you could lead them through a requirements review now, with the goal being to remove any steps they will not ultimately need, thus making the user's job simpler. How else could you offer to automate this process to further speed up their return handling?

Other Transactions and Custom Transactions

NetSuite's flexibility is well known, and for good reason. When it comes to the kinds of miscellaneous transactions covered in this chapter, no two companies will use them the same way, and some companies' businesses are so unusual that they need custom alternatives to the standard transactions. So, in this chapter, we'll explore some of the many other transactions available out of the box or via special configurations.

Revenue recognition, which is part of NetSuite's **Advanced Revenue Management (ARM)** feature set, is especially important. I've mentioned it a couple of times before, and we'll dig into the topic in more depth here.

In this chapter, we will cover the following topics:

- Using work orders and assembly builds for manufacturers
- Managing revenue recognition with ARM
- Using cases and issues for customer support
- Creating custom transactions for special sales and purchases

Your contacts for this work will vary for each of the special transactions described here. By understanding how and when to use them, you'll know how to guide your client to use these transactions when the need arises.

Using work orders and assembly builds for manufacturers

Generally, manufacturers build the finished goods they sell from components they purchase (that is, raw materials). They purchase the components from their suppliers or might make some of those themselves as well. They process those in various ways (which is work they need to track since it represents an expense), and the result is the product they sell, plus some waste/scrap that's left over (very frequently). Whether they're making food products, craft items, or clothing, most manufacturers follow one of these two patterns.

Here are the two most common manufacturing work processes we track in NetSuite:

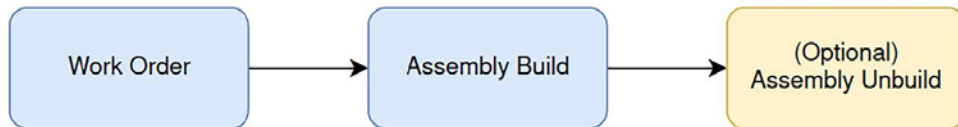


Figure 16.1 – Workflow 1 with work orders and assembly builds

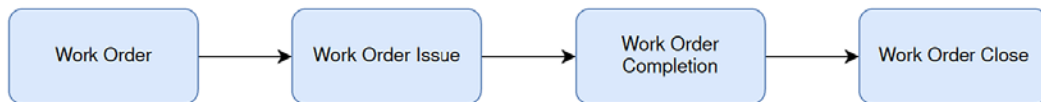


Figure 16.2 – Workflow 2 with Work In Progress (WIP) and Manufacturing Routing features in use

Businesses that manufacture products usually have extra requirements around that process. They want to track their steps and costs, especially throughout the process, to ensure that they stay in the black. NetSuite offers a set of transactions for this purpose, so let's explore each of those (that is, making food products, craft items, or clothing).

Assembly items

We need to define both the component items and the finished goods items in the system. The components will typically be set up as inventory items, but they can be assemblies as well. The finished goods are set up as **Assembly Items**. An assembly item in the system can also be lot-numbered or serialized when needed. NetSuite understands how these items will be used and includes a lot of features to handle and account for them correctly.

For example, if a company makes bakery items for distribution within its local region, it will define inventory items for salt, flour, baking powder, and so on. Then, it will define assembly-type items for the cakes and muffins it makes from those ingredients. For each of these items, these businesses will typically track their costs both from the suppliers and their manufacturing processes.

Assembly items are commonly brought together into what's known as a **Bill of Materials (BOM)**. This is a printed output we can generate for any assembly item and its components, listing the items used and their quantities. We can print a BOM from the **Item** screen or a work order transaction. Users can also access another screen for this purpose, called **Bill of Materials Inquiry**. When the basic BOM included in the system isn't enough for some manufacturing clients, we have an advanced BOM with features such as multiple BOMs per assembly item and multiple versions of each BOM.

Work orders

Whenever a business decides to combine the raw materials (that is, components) for an assembly item to make new finished goods, it will start the process with a **Work Order**. This is a native NetSuite transaction that's used to tell the workers out in the plant, bakery, or other location to make new products.

A work order includes fields that allow us to select an assembly item in the header, set quantities and locations, and inventory details. This automatically includes the component list for the assembly. The quantities on those lines can be adjusted when that's needed and when the transaction is saved, as can its status, indicating when the workers should start. Some companies need a more fine-tuned process for this handover, in which case we can use the WIP/Manufacturing Routing features described shortly.



For accounting purposes, please note that work orders in NetSuite do not post to the GL, since they represent an order only; builds and so on do, though, since they represent finished work.


Assembly builds

Once a work order has been *released* to manufacturing, steps are followed to create the finished goods in the plant, bakery, warehouse, and so on. If all the business needs to know is when that processing will be completed, then the next transaction it'll use is called an **Assembly Build**. This is where we record what was created, since that might not always be exactly what we sent out on the work order.

For instance, let's say there is waste or scrap material left over in many processes. Woodworkers, for instance, will usually have some small pieces of scrap wood left as they make furniture from a sheet of plywood.

Here is an example of an assembly build:

 **Assembly Build** 

Save 

Cancel

List

Search

Customize

More

Primary Information

REFERENCE #
To Be Generated

ASSEMBLY *
Badminton Set

REVISION

BUILDABLE QUANTITY
15

QUANTITY TO BUILD *
50

UNITS

PROJECTED VALUE
2,150.00

SERIAL NUMBERS

BIN NUMBERS

DATE *
8/30/2021

EXPIRATION DATE

POSTING PERIOD
Aug 2021

MEMO

Classification

SUBSIDIARY *
Parent Company

DEPARTMENT

CLASS

LOCATION *
Main Warehouse

Components

Communication

Customize

COMPONENT	QUANTITY	QUANTITY ON HAND	UNITS	SERIAL/LOT NUMBERS	BIN NUMBERS
100	100	30			
101	50	40			
102	50	40			

Figure 16.3 – An assembly build transaction nearly completed

Assembly builds are used to record all of this and remove the component items from the inventory while the assembly items are added to it. Forms like these require us to select **SUBSIDIARY** first and then **ASSEMBLY**, **LOCATION**, and the rest of the fields. Remember that with builds, the costing method is already set on the assembly items, so double-check that setting if you're concerned about getting the right GL impact at this stage.

WIP/Manufacturing Routing

When a manufacturer needs to control its process even more tightly, it enables extra features in the system known as **Work In Progress (WIP)** and **Manufacturing Routing**. WIP adds transactions to the system, allowing companies to break down the work process into three steps: **Work Order Issue**, **Work Order Completion**, and **Work Order Close**. Having these additional transactions allows the manufacturer to track its raw material inventories as more than just a work order in the assembly build process. It will use each step in NetSuite to indicate which materials have been consumed as each step

is processed. So, for instance, if it makes cured hams from raw pork products, and the hams need to cure in one stage for several weeks, it will be able to determine at any moment how much money it has tied up in each step in that process. Those costs can be included in its period closes and reports. When it only takes a day to make a product, the WIP records are less likely to be useful to the business.

Manufacturing routing takes this idea of breaking down the steps even further. When a company needs records not just saying, “*We started the manufacturing*” and “*We finished,*” but also which station in the process each work order is currently at, then they can set up routes in NetSuite and use the route steps to record each activity and adjust the costs, if needed, as the work is performed. This is more complicated than just a paragraph can properly describe, so check out the Help pages on *Manufacturing Routing* if you ever need to include this level of detail in a manufacturing process.

From the explanation provided so far, I hope you can see how NetSuite delivers what manufacturers need natively, but we do sometimes need to augment some of those features with SuiteApps and other third-party solutions. Third-party vendors offer other advanced apps as bundles that we can add to an account when we need them. They’re listed in the SuiteApp Marketplace, as always.

New features in 2025.1 – Work Instructions and Traveler, and Manufacturing Scheduler SuiteApps

If you are working with a client who intends to use all of the advanced work order features, including WIP and Manufacturing Routing, then it makes sense to also check out this new SuiteApp. Provided by NetSuite, the **Work Instructions and Traveler** SuiteApp includes features allowing clients to define their Traveler document in advance, based on templates, and then to generate those as they release orders to their manufacturing centers. The Traveler document can include custom instructions for the assembly items being produced, and those can include images, documents, or even helpful videos for the operators to view. Check out the NetSuite Help page titled *Work Instructions and Traveler* for more information.

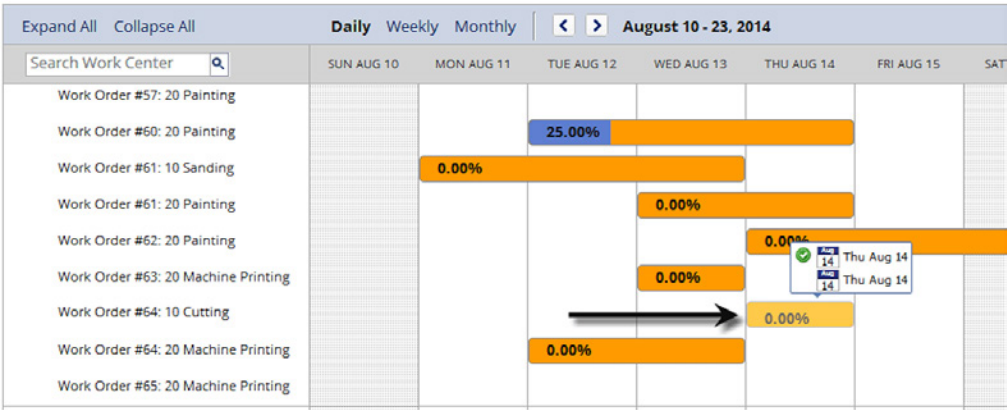


Figure 16.4 – An example of a Manufacturing Scheduler

Also new for use with advanced manufacturing clients is the **Manufacturing Scheduler** SuiteApp. With this feature added to your NetSuite account, manufacturing managers can keep track of and schedule operations in a visual interface for all of their operation centers. On the new **Manufacturing Scheduler Dashboard**, managers can see all activity in one screen, organizing their data by work order and/or work center. The screen is customizable and allows for filtering on the fly, to make sure everyone is productive, and the work gets done on time. See the Help page titled *Manufacturing Scheduler* for more info.

Next, we'll cover how NetSuite helps companies with special revenue recognition requirements.

Managing revenue recognition with ARM

Revenue recognition (rev rec) is an important topic for any company that enters into contracts of any sort with its customers. The financial industry has settled on standards and practices for accountants to follow regarding recognizing revenue from these arrangements, so that investors, governments, and others can confidently compare financial results from all businesses. In the past, without these standards, it was difficult to know what made one company stronger or more successful than others, based solely on its financial reports.

Companies that sell all of their products or services in a very short-term time frame don't have this problem, since they both bill for their products and receive payments in a relatively short time. However, companies that enter into years-long contracts with their customers, such as cable TV providers or software publishers, must follow these rules so that their revenue reporting is predictable and accurate.

For example, think about any contracted service you might have signed up for, where the provider delivered it to you over a year or longer. That process might look like this, in terms of the transactions we use to track the contract in NetSuite:

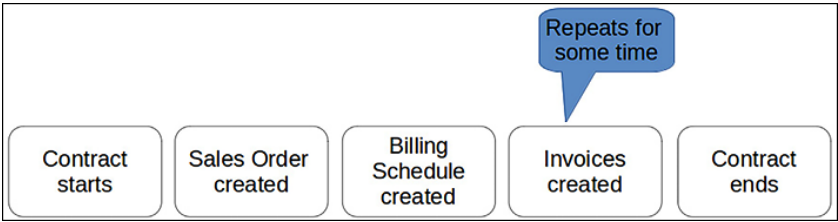


Figure 16.5 – A standard sales order/invoice process for a contract

Companies that charge a flat rate for their service, as a monthly charge, for instance, typically have a fairly simple time getting this set up in NetSuite. But companies that sell a general service in advance and then charge customers based on services they receive over time have more work to do. Think about a legal services firm that requires customers to sign up for annual contracts but then charges each customer for the specific legal assistance they receive. They might have a flat rate for the basics, and they might charge extra for things such as document-handling or court filing fees. They need to be able to sell the client one thing (the annual service) but then also charge them for the specific services they received.

At the beginning of the contract's term, the company knows what the full value of the contract will be (excluding those additional charges), but it can't report all that revenue as if it has already been received until it is paid for all of the services (or whatever the conditions of the contract call for). It would be misreporting its real income if it did. It's for this reason that the standards bodies (such as **Financial Accounting Standards Board (FASB)** and **International Accounting Standards Board (IASB)**) have gotten more and more strict about rev rec rules – to control this reporting process. This is a fairly complex topic and requires quite a lot of accounting expertise to get right, but in a nutshell, rev rec rules only tell businesses how to report revenue as they deliver the goods or services promised in their contracts over time.

I am not an accountant and have not received the proper training to cover this topic in depth. So, I'll be covering the main NetSuite features you should know about when implementing a client who has to follow rev rec rules in this section, but you should always ensure you're receiving expert guidance from a trained accountant when performing these steps for a client.

For more information on this, outside of the NetSuite Help pages, check out the FASB website: <https://www.fasb.org/>.

In NetSuite, we have a set of features you should be aware of when implementing any software, services, or other types of businesses that need to follow rev rec practices. We'll cover these in the following subsections.

Advanced Revenue Management

This is a comprehensive module we can enable in an account when we need to take advantage of NetSuite's Advanced Revenue Recognition workflows, revenue arrangements, rev rec rules and plans, and Fair Value Price tracking.

The ARM feature has a cost associated with it, so a NetSuite sales representative must enable it in an account for us. It's generally the kind of thing we want to know about from the start and plan to include in our implementation. It can be enabled later if a business starts out using *classic* revenue accounting standards but then needs to adopt rev rec later on; however, this is generally difficult and time-consuming. Review *Chapter 6* for more on gathering these requirements.

Since NetSuite 2022.2, this feature has changed and is now split into two features, so we can enable and use them separately or together. The first is called **Advanced Revenue Management (Essentials)**, which includes *Revenue Arrangements* and *Revenue Plans*. The second is called **Advanced Revenue Management (Revenue Allocation)**, which adds fair value pricing, range-checking, and fair value formulas. With this change, the existing features don't change for accounts already using them, but new clients can choose which options to enable, per their own requirements.

Revenue recognition rules

When you need to follow a rev rec process for some or all of a business's sales, you need to define the rules its rev rec should follow. The accounting standards mentioned in the preceding section dictate most of this, but the system allows for some flexibility in the rules' application. This is what a rev rec rule record looks like, when partially filled in, as an example:

Revenue Recognition Rule

Save

Cancel

Primary Information

NAME *

Rev Rec Rule 1

RECOGNITION METHOD *

Straight-line, by even periods

RECOGNITION PERIOD

PERIOD OFFSET

0

START OFFSET

0

☐ INACTIVE

INITIAL AMOUNT

80.0%

TERM IN MONTHS

TERM IN DAYS

AMOUNT SOURCE *

Event-Percent based on quantity

REV REC START DATE SOURCE *

Event Date

REV REC END DATE SOURCE *

Event Date

END DATE CHANGE IMPACT *

Update Remaining Periods Only

REFORECAST METHOD

Last Period

RECALCULATION ADJUSTMENT PERIOD OFFSET

NO.	PERIOD OFFSET	AMOUNT PERCENTAGE *
1	1	10.0%
2	6	50.0%
3	12	40.0%

✓ Add

✕ Cancel

+ Insert

Remove

Save

Cancel

Figure 16.6 – An example of a Revenue Recognition Rule record

Each rule we create indicates a specific time frame for which the rule will be in effect, any delay at the start before revenue should be recognized, and so on. We apply the rules we create to revenue plans and those are associated with individual sales. We can import rev rec rules if we need to create them in bulk, just before going live, for instance.

Revenue recognition plans

A **revenue recognition plan** is generated by NetSuite every time an item that is set up for rev rec requires one, based on the rev rec rules. That happens when the **revenue arrangement** (which we will learn about next) is created, whenever the item is fulfilled, or when the item is billed, depending on the settings used.

Plans can be created automatically by the system, or you can reserve that as a manual process (which might be useful in some rare cases). Plans tell NetSuite how much revenue to recognize and in which accounting periods. This actually happens when a set of journals is created, per the plan.

Revenue arrangements

Arrangements are non-posting transactions that NetSuite typically creates for us when a rev rec plan is in place. Similar to a billing schedule (which controls the schedule that invoices will be created on), a revenue arrangement tells the system when revenue should be recognized, so it usually becomes part of the period close process. We can view these records on a sub-tab on the related sales transaction (a sales order, a standalone invoice, or a cash sale), and users with the right permissions can edit them if needed. They can also be generated via manual processes for clients who need to take more control.

Revenue allocation

For some companies, in addition to rev rec rules and arrangements, we need to control the process of allocating revenue across sets of items or charges per a business's requirements. For example, if a company sells a **software-as-a-service (SaaS)** product, including user licenses and software maintenance delivered over time, not all of those items will be sold for what the accounting standards refer to as their *fair market value*. NetSuite gives us a way to allocate the full sales price the customer is paying, spread out over time. Allocation is frequently complicated by what are known as *carve-outs* and *carve-ins*, such as returns for items, and more.

Amortization

Expenses can be amortized in NetSuite as well. When we enable this feature, we're setting up various transactions, which will allow the business to spread the cost of some expenses over time. For many businesses of all types, this helps a lot since the costs of some items are spread out over many months or years, just as revenue can be. For example, if the company signs a contract with a service provider, it might want to record the total it will owe at the beginning but then defer some of the expenses to each month in the future. We can set up **amortization templates** and schedules in an account to define which items/services this deferral applies to and how it should happen.

Here is an example of an amortization template, ready to be used with expense transactions:

Amortization Template

List Search

EditBackActions

NAME

12 month, by trans. date

TYPE

Standard

METHOD

Straight-line, by even periods

TERM SOURCE

Transaction Date

DEFERRAL ACCOUNT

Deferred Expense

CONTRA ACCOUNT

Savings

TARGET ACCOUNT

Miscellaneous Expense

AMORTIZATION PERIOD

12

PERIOD OFFSET

0

START OFFSET

0

RESIDUAL

INITIAL AMOUNT

☐ INACTIVE

RecurrenceSystem Notes

ACCOUNT	PERIOD OFFSET	AMOUNT
Purchases	0	1,001.00

Figure 16.7 – An amortization template, set up for 12 months and evenly allocated

These ARM features seem daunting at first, but in practice, you can learn about them fairly quickly and help clients learn how to use them with some guidance. The NetSuite Help system contains a lot of information, including videos you can learn from.

Next, we’ll explore another side of the system: customer support.

Using Cases and Issues for customer support

Since NetSuite wants to be the central place where each client runs their business, it makes sense that the system offers features for customer support organizations. The main features worth knowing here are **support cases** and **issues**. Cases are used to track communication with a customer and can be created either by the customer or the client’s **customer service representatives (CSRs)**. A customer can typically create a case via email or by completing a web form. The client’s employees can manually create cases whenever they need and can also use the **CSV import** feature to create cases in bulk, as is common if NetSuite and another system are kept in sync.

Cases can be used for many purposes, but generally, they should be related to customer support. For instance, their most common use is to track questions and reports of problems from customers with products or services they’ve purchased. But we’ve also seen clients use cases to track internal employee concerns as well, such as questions about benefits or pay slips. (This is why the **Company** field on the **Case** form includes both customers and employees by default.) The system includes a set of starter statuses, priorities, and types for cases, but those can be easily customized to suit a business’s needs. We can also create custom case profiles so that companies with multiple subsidiaries can have separately branded messages attached to the cases for each part of the business.

Support cases can be tied to transactions as well, so that, for instance, a customer who placed an order can contact Support and ask questions or make changes later, before the order is shipped. The CSR who opens the case might not have the sales order number at first, but they can link their case to the order later via the **Related Records** tab. Just be careful with this, since, out of the box, the system allows any transaction to be linked to any case. Some clients want a restriction so that CSRs can only assign transactions to cases for a matching customer.

Issues are related to cases and are typically used to report problems the company needs to fix. For instance, a software company might use issues to report defects to the product's developers so that it can prioritize and resolve them. A maker of coffee machines might use issues to track customer enhancement requests and physical product defects internally. Issues can be created by customers if **Customer Center** is in use or by the client's employees. They can be linked to cases and have customizable lists for things such as status, priority, severity, and more.

This is what the default **Issue** screen looks like:

The screenshot displays the 'Issue' screen with the following fields and sections:

- Header:** 'Issue' title with a search icon, 'Save' button, and 'Cancel' button.
- Primary Information:**
 - ISSUE NUMBER: To Be Generated
 - ISSUE TYPE: Problem (dropdown)
 - PRODUCT TEAM: (dropdown)
 - ASSIGNED TO: (dropdown)
 - ISSUE STATUS: Submitted (dropdown)
 - SEVERITY: S1 - Blocking (dropdown)
 - PRIORITY: P1 (dropdown)
 - SHOWSTOPPER: ☐
 - ABSTRACT: Customer reports Issue with Items purchased. (text area)
- Problem Information:**
 - ITEM: (dropdown)
 - SOURCE: Internal (dropdown)
 - REPRODUCED: Test Environment (dropdown)
 - PRODUCT: Orange Peeler (dropdown)
 - MODULE: (dropdown)
 - TAGS: <Type & tab for single value> (text area)
- Details:** A tabbed interface with 'Details' selected, showing 'NEW DETAILS' (text area). Other tabs include 'Communication', 'Versions', 'External', and 'Related Records'.

Figure 16.8 – The default Issue screen and its fields

Using cases and issues in NetSuite makes sense for many small and medium-sized businesses, since the customer support data can easily be linked to transactions. It can also be tracked via the same types of reports the rest of the business uses. There are a lot more support-related features here to explore, including the creation of a knowledge base and solutions, and so on. The NetSuite Help pages contain all the necessary information regarding this under the *Support Management* heading.

These support-related features and records can be a big help to any client who has a support team and wants them to share the ERP system with accountants, sales teams, and other users. Support cases and issues will get them most of the way there. Clients who choose to add knowledge base articles and solutions can complete the ERP system support in most cases.

For the last section of this chapter, we'll take a look at the system's custom transaction features and learn when they are most useful.

Creating custom transactions for special sales and purchases

Since we're talking about special types of transactions in this chapter, we should also take a moment to talk about a little-used but still very helpful feature called **Custom Transactions**. NetSuite allows us to create a new, custom type of transaction whenever we need, but we generally only do so when the client has requirements that can't be met with the native transaction types. For instance, if a company wants to track a special type of sales order (SO), we will usually just add a custom list field to SOs, allowing them to be categorized as needed.

In some cases, though, it makes sense to completely segregate the transactions into their own, new custom type. Think about a business that offers complex financial transaction processing or legal services, for instance. They might balk at the idea of tracking things such as a special kind of long-term debt write-off or arbitration services on an SO. Or maybe a company wants to consolidate their invoices for their clients into something else and then handle those transactions differently from native invoices. Creating a custom transaction in these cases makes sense and is not too difficult.

Once we enable the **SuiteBuilder Custom Transactions** feature, we can define the type based on several **styles**. NetSuite offers a **Basic** style, a **Journal** style (with debit and credit lines just like a journal entry), a **Header Only** style (which requires the creation of a custom GL plugin script), and **Sales** or **Purchase** styles, for when the transaction should act almost like a sales order or purchase order.

In one project I worked on, a customer wanted to take SOs from customers, as most companies do. But then they wanted to consolidate the items from those orders into a special transaction they called *Outbound Shipment*. In their business, this was critical, since their orders tended to be large and include a great variety of items, and shipping those orders involved combining items across orders, based on ever-shifting product availability. The rule we put in place was that shipping could only happen from one of these *Outbound Shipment* transactions. We created a custom sales-style transaction type for this purpose and set up customizations (scripts, workflows, and integrations with the warehouse management system) for those. This combination of native and custom features allowed the client

to track their sales and shipments in two separate lists in NetSuite, and this approach made the most sense to the managers and users who had to work with the system. Having a custom transaction type also gives users more control over access permissions and reporting abilities.

Keep this Custom Transactions feature in mind for those special cases that arise every so often with clients whose businesses don't fit into the native transaction use cases. It's also handy to use a custom transaction when it feels like you're stretching the normal meaning and use of a native transaction.

Summary

This chapter showed you how NetSuite gives businesses so much room to stretch out and make their accounts unique to their processes, while also adhering to global accounting standards. You should now know how NetSuite extends its core financial, purchasing, and sales features to cover as many other related business processes as it possibly can. Work orders and their related transactions give manufacturers the same access to quality data as other businesses get with the basics. Revenue tracking records and transactions give software and services businesses a leg up on their competition. Support cases and issues give the customer service teams within a business a place to track their work, and custom transactions allow us to extend the system even further.

In the next chapter, we'll take a look at how all this data inside NetSuite can be put to use, helping clients make decisions and plan for the future.

Self-assessment

Take a minute to see whether you can apply what you learned in this chapter to a few situations you might run into in the field. Think about how you would handle each of these cases. There is no single right answer for most of these:

1. Your client is not using any of the manufacturing features yet, since they are primarily a distributor of packaged consumer goods. They ask you for help to track a multi-step process they follow on the warehouse floor, where items are bundled together from a shortlist of small components into gift baskets. How do you help them decide whether they need work orders and assembly build transactions for this, or whether they could get along with something simpler?
2. *ABC Co.* has a fleet of trucks it uses for deliveries, but occasionally the demand is high enough that it has to rent trucks for a day or two as well. It's contracted with a local company to rent short-term trucks when it needs them. It will pay a flat rate per month for access to the vehicles and then a per-mile cost plus gas whenever it uses one. Which NetSuite features can help it manage and track these expenses?
3. During implementation, your client learns about the NetSuite **Knowledge Base** feature, and they ask for help setting it up and getting all of their existing articles from another system into their NetSuite account. What's the right level of assistance in cases like this? For instance, should you offer to convert the data for them into the NetSuite format?

4. Your customer wants to *tag* certain transactions as being part of a beta program they're running with certain customers. They want to be sure that every transaction that's created within this program is *tagged* this way; this will allow them to easily differentiate between the results on financial and other reports. Is this a good use of custom transactions, or is there another NetSuite feature that might help the company more?

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search for this book by name.

Note: Keep your purchase invoice ready before you start.



Analytics, Reports, and Data Exports

By the time you start to really think about reports and exports in your implementation projects, you should have already helped your client enter many types of data into their NetSuite account. Always remember, though – people don't use computers because they like looking at numbers, dates, names, and so on; they use computers to gain insights from data.

In this chapter, we will explore all the ways a client might want to use their data for analysis, decision-making, and feeding other processes outside of the system, and how you can help them achieve this.

In this chapter, we will cover the following topics:

- Using searches and views to find information
- Creating reports for in-depth analysis
- Utilizing datasets and workbooks for deeper analysis
- Creating exports with SuiteAnalytics Connect

You might need to work with a wide variety of the client's users on these features since everyone has data in the system and most need to find it and use it for a range of purposes. As always, these users should start with some introductory NetSuite training, but then if some of them can take one or more of the SuiteAnalytics courses too, that's a bonus. The point in the implementation project at which you train the users on these things will vary. Some projects require the client's users to know how to get hands-on with searches very early, but for most, this can wait until they've really started to define their account and capture data in it.

Using searches and views to find information

Nine times out of ten, when a client asks a question such as, “How can I find all the X that have Y?”, we point them to searches in the UI. Keep in mind that searches are very powerful, and it’s only occasionally necessary to use any other more involved features.

We create searches for ad hoc needs, such as when we need to find an example of an item that has been set up for drop shipments for a demo, and we create them when we have a recurring business process that needs a specific list of things. An example of the latter is when a company wants to find all of the sales orders they ended up losing money on. (This should never happen, of course, but by the same thinking, you want to know if it ever does.)

Global search

The simplest search that users can perform uses the global **Search** field, right at the top of every screen in NetSuite. This simple-looking field is more powerful than many realize:

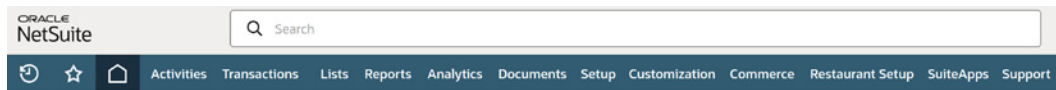


Figure 17.1 – The humble but very useful global Search field

Here are my favorite tips for getting the most out of global search:

- **Alt + G:** This is the keyboard shortcut you can use from any NetSuite screen to jump to the global **Search** field.
- **Custom records:** You can define custom records to be included in or excluded from global searches. This is helpful for records that a group of users needs quick access to, such as warranty records or customer serial numbers. It’s also a good idea to exclude records you’re trying to keep most users away from, such as a table that’s only used by a script.
- **Inactive records:** By default, records marked as **Inactive** are not shown in the results of a global search. You can change that in **User Preferences** (go to **Home | Set Preferences**) or you can add a plus character (+) to the end of your search terms: for example, `item: old product 100+`.
- **Help:** You can search the Help system from the global **Search** field. Just use the `help:` prefix before your terms: for example, `help: inventory adjustment`.

In NetSuite’s 2023.1 update, the system now includes all of these types of items in search results from the global **Search** field by default:

- **Menu item:** You no longer have to remember how to access certain features; let the system find any page for you.
- **Fields on the current page:** View a sales order, customer, and so on, and then use the global search to find any field on the page. This is really handy for forms with many subtabs.
- **Everything else:** The global search still allows you to find all of the records in your account, just as it has in previous releases.

Saved searches

Saved searches are great for when you want to get more advanced regarding the criteria to be applied to a set of records, or when you're using a search such as a report, where a user or a group needs to run it more than once. Learning how to use all of the features of searches will make a difference for your users. Getting them just the results they're looking for – or better, helping them learn how to do this for themselves – can make you the hero of many otherwise troublesome situations. If you need more info on anything covered here, you can review the Help section on *SuiteAnalytics | Search*.

Here are the saved search limitations and features every NetSuite business analyst should know about:

- **One record type:** Searches always focus on a single record type, but the system's architects knew we'd need some flexibility in that, so we are allowed to select **Entity** and **Transaction** as the record types when we want to include more than one type of those records in the search's results.
- **Expressions in criteria:** Don't let these throw you; they're not hard to use and are frequently required to find just the right set of data you're looking for. I find it helps to try to think about sets of records (those you want to include and those you don't want to see), and then to picture a Venn diagram with those sets of data. Do they overlap, and if so, how? For instance, if you know you want to include any sales orders that either have one type of item or have a total of more than a certain amount, you will need to use expressions with parentheses to find them.

Here's what an expression looks like in a saved search:

Saved Transaction Search

ListSearch

Save & Run

Cancel

Preview

Actions

Search Title *

Open Transaction Search

ID

☐ Public

☐ Available as List View

☐ Available as Dashboard View

☐ Available as Sublist View

☐ Available for Reminders

☐ Show In Menu

Criteria

Results

Highlighting

Available Filters

Audience

Roles

Email

Audit Trail

Execution Log

Search Title Translation

Use this tab to specify criteria that narrow down your search.

☒ Use Expressions

Standard	Summary
Not	Parens
	(
	Type
	Status
	(
	Type
	Amount (Transaction Total)

☐

✓ Add

✕ Cancel

+ Insert

Remove

Save & Run

Cancel

Preview

Actions

Figure 17.2 – An expression used for a search's criteria

- **Mainline:** This is an odd term, for sure, but in the system, a transaction search that only focuses on the mainline will not include any line-item details; a search with **Mainline** set to **False** forces NetSuite to include line-item details where it might not otherwise. This is important since some of the fields we can select from in the columns have the same name in the header of their transactions as on the lines. This is true for **Amount** and **Location**, for instance, so choosing the right value for the **Mainline** setting can make the difference in getting just the right set of results you're looking for.
- **Summarized results:** Sometimes, we don't want to see every line item on a set of transactions (for instance), but we need to summarize them somehow. That's when this feature comes in handy. For instance, if the warehouse people need a quick idea of how many sales orders that are ready for shipping on a given day have special order items, we can create a search with summarized results to show them the sum of the quantities of those items instead of hundreds of detailed lines. We can also summarize results by grouping them, counting them, and getting minimum and maximum values for them.

- **Highlighting:** This is a great feature, so long as it's used correctly. Nobody will be able to work with a set of search results where every line has a different font and background colors, or where some rows are bold and some italicized. Instead, my guideline is to use highlights very sparingly to make only the most important records stand out from the crowd, and only in cases where the highlight indicates that an action is needed from the user.

The Oracle NetSuite-provided training course called *Business Intelligence with Saved Searches* (<https://mylearn.oracle.com/netsuite/learning-path/business-intelligence-with-saved-searches/126628>) offers even more powerful user tips to help you make the most of saved searches. See this page on the NetSuite website for a list of all of the available educational offerings: <https://www.netsuite.com/portal/services/training/suite-training/catalog.shtml>.

Speaking of searches, don't forget that we can customize almost any list of records in an account with what's known as a **view**. These are searches that are enabled for use as a view, and they can include most of the features mentioned previously. We briefly talked about these before, and it's worth mentioning here that these are very important for both native and custom record types. They help users find the records they need to work with and get their jobs done. A good view with the right filters applied and also available on the screen for the user to select from is helpful to new users of the system.

Searches and views are used to convey data to users in ways they can use every day. Whether they're a customer service rep with a customer on the phone or a CEO who needs to know how sales are looking this month, you can help them solve the problems they face by knowing all the ins and outs of these features.

Creating reports for in-depth analysis

NetSuite offers another feature that helps with in-depth analysis, and that is where **reports** shine. A new account comes with a set of predefined reports out of the box for many areas, including finances, items and inventory tracking, accounts payable and receivable, and much more. These native reports can mostly be customized, but not infinitely. For instance, the balance sheet financial report can be customized to include almost any period you want it to display, but it cannot be made to show item inventory levels (not that you would ever want that, of course).

The system does allow us to customize most of the native reports, though, and the options we have on that screen are generally similar, though some options will only be available when they make sense for the report you start from.

The following is the UI for customizing a financial report, such as a balance sheet:

The screenshot shows the 'Financial Report Builder' interface for a 'Custom Balance Sheet'. At the top, there are tabs for 'Edit Layout', 'Edit Columns', 'Filters', 'Sorting (Optional)', and 'More Options'. Below these, the 'Name' field is set to 'Custom Balance Sheet', the 'Subsidiary Context' is 'HEADQUARTERS (Consolidated)', and the 'Layout' is 'Custom Balance Sheet Layout(US)'. There are 'Edit Layout' and 'Preview Layout' buttons. On the left, a 'Layout' sidebar shows a tree structure: 'ASSETS' (with sub-items 'Current Assets', 'Total Current Assets', 'Fixed Assets', 'Other Assets'), 'Total ASSETS', 'Liabilities & Equity' (with sub-items 'Current Liabilities', 'Total Current Liabilities', 'Long Term Liabilities', 'Equity', 'Total Equity'), and 'Total Liabilities & Equity'. The main area is titled 'Header Row: ASSETS'. It has a 'Header Label' field set to 'ASSETS' with a 'Display Row' checkbox checked. Below it is a 'Summary Label' field set to 'Total ASSETS' with a 'Display Row' checkbox checked. There is a 'Child Of' dropdown menu. The 'Display' dropdown is set to 'Expanded'. Under 'Summary Row Calculation', the 'Dynamic Total' option is selected, with the description 'Sum all rows within this header.' The 'Apply a Formula' option is also present, with the description 'Calculate the summary using the formula builder below.' The formula builder shows '[Total ASSETS] =' followed by an empty input field. Below the input field are buttons for 'Add', 'Cancel', 'Insert', and 'Remove'.

Figure 17.3 – The customize screen for the balance sheet report

With these options, we can change the general layout of the report and what will appear on the columns and rows of the report’s contents. We can choose how the content will be summarized on the totals rows as well.

If the native reports don’t get us what we need, we can always create a new custom report from scratch, based on the report types available in the account. Remember that all reports have permissions, and users will need to have the right permissions to be able to both create and view any new report you create. In a OneWorld account, many reports require us to make a subsidiary choice, so our view in the report will be restricted to just that selected subsidiary. We can do that by going to **Home | Set Preferences | Restrict View**.

When we need to start a report from scratch, we must look at the top of the **Reports** menu in the UI. First, we must choose between creating a **New Report** or a **New Financial Report**. The **New Financial Report** option allows us to create new balance sheet, cash flow, income statement, or cash statement reports. The other option is for all other types of reports. There is a long list of report types in that second list, which will vary depending on the features you have enabled in each account. We can use this feature to create reports for the support personnel on things such as cases and issues. Alternatively, we can help the sales team keep track of items, as well as inventory and sales figures.

There are also reports for system administrators for tracking things such as integration use and web store sales metrics.

There are a lot of things we can customize about each report type in the system, so I usually recommend that people take the offered Oracle NetSuite training on this topic if they're going to spend any real time creating and/or using this feature. The UI is different from every other analytic feature in the system, and it's not the most user-friendly, in my experience. Having an experienced person guide users through the process is usually key to their success.

Let's take a quick tour of this process now, though, by customizing a relatively straightforward report: **Transaction Detail**. You can create your own copy of that report by going to **Reports | Financial | Transaction Detail | Customize**. Once there, you'll see a screen like this:

Report Builder | Custom Transaction Detail More

Edit Columns | Filters | Sorting | More Options

Name: Custom Transaction Detail

Search Fields

Add Fields

Add Formula Field

Transaction

Report Pre

Type	Date	Document Number	Name	Memo	Account	Clr	Split	Qty	Amount
Type 1	5/25/2025	Document Number 1	Name 1	Memo 1	Account 1	Clr 1	Split 1	1	\$50000.00
Type 2	5/25/2025	Document Number 2	Name 2	Memo 2	Account 2	Clr 2	Split 2	2	\$40000.00
Type 3	5/25/2025	Document Number 3	Name 3	Memo 3	Account 3	Clr 3	Split 3	3	\$30000.00
Type 4	5/25/2025	Document Number 4	Name 4	Memo 4	Account 4	Clr 4	Split 4	4	\$20000.00
Type 5	5/25/2025	Document Number 5	Name 5	Memo 5	Account 5	Clr 5	Split 5	5	\$10000.00
Total									\$150000.00

Move Left | Move Right | Remove Column

Transaction: Amount (Gross)

Column Label: Amount

Summary: Summary

Drop Decimals: ☐

Divided by 1000: ☐

Display Negative Numbers: Parentheses example: (123)

Negatives in Red: ☐

Add Grand Total: ☒

Add Running Balance Column: ☐

Add % of Total Column: ☐

Figure 17.4 – The screen we use to customize the Transaction Detail report

Here, you can change quite a lot about the report. The **Edit Columns** screen allows you to change the column names and order and other properties. Next, the **Filters** tab allows you to add filters, so you only see the rows of transaction details you need. For instance, you might want to only see results for one subsidiary and its children. The **Sorting** screen allows you to control the order in which the results will be shown, and the **More Options** screen has other settings, such as whether you want to see zeros or blanks in the details, and whether sections should be expanded by default. When you're customizing a report like this, be careful to use the **Save As** option when you're done editing, so you don't overwrite the default report.

Once you have set up a report, working with it is straightforward. You can view it any time you need, share it with other users, and export it to a variety of formats, including PDF, Word, and Excel. You can print it and send it via email as well. Another useful (but often forgotten) export option you can enable for most reports (other than the native financial statements) is **Microsoft Excel Web Query** and its files. These IQY files are definitions of the report you can download and then view in Excel. Each time you do that, you refresh the data defined in that report by pulling it down from NetSuite. For many companies, this can be very handy, since they can share the Excel file to select groups of co-workers, without necessarily having to train them on NetSuite.

Reports are great when a more polished set of results is needed; they can look nicer than searches, for instance, especially on the financial side of the house, where standards dictate what these reports should look like. Now, let's move on to one of the newer features in SuiteAnalytics – workbooks.

Utilizing datasets and workbooks for deeper analysis

Most of the time I've worked with NetSuite, I've faced a real challenge when it came to searching for and reporting on data in the system. That has to do with what are called **joins**. If you think about sales orders, they are joined to the customers list, by way of the **Customer** (or **Entity**) field on the form. Customers are also joined to other lists, such as currencies and credit cards. If you want to get a list of sales orders that contain certain customers, but only those who have more than one credit card, you can't use saved searches. That's because they can only "see" one join per search. Looking into that credit card list on the customer record, the transaction would require NetSuite to make a second join, and that's beyond the capabilities of saved searches. In the past, we used to solve this by doing more than one search, and then somehow stitching the results together, with help from the users, via exports to Excel or a script.

Within the last few years, though, NetSuite has introduced two new features – SuiteAnalytics **workbooks** and **datasets** – that are like saved searches but more powerful, and they do not have this restriction on joins between record types. This is NetSuite's answer to the people who said the system doesn't include any useful business intelligence tools, and it's great once you get used to working with it. We can access these features via the **Analytics** option on the main menu bar in the UI.

Here's what the default analytics screen might look like in a typical account, with a mix of custom and predefined workbooks on display:

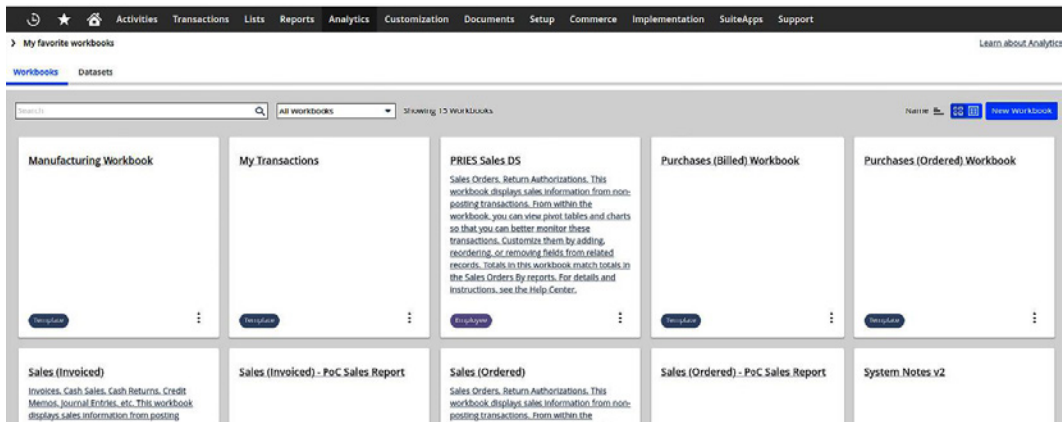


Figure 17.5 – Example of the analytics screen in the NetSuite UI

With a dataset, we can bring together pretty much any related data we can find in the account. Building a dataset includes selecting records and fields and then applying filters to them. Since datasets can contain templates they're derived from, every new NetSuite account should come with a few examples to get you started. We can explore those, save copies, and tweak them as we need, or we can always create a new dataset from scratch. To do that, select **Analytics** from the menu bar, then the **Datasets** tab in the upper-left corner. Then, click the **New Dataset** button, and select which record you want the dataset to focus on in your account.

Here's what the screen for creating a new dataset looks like when you select **Support Cases**:

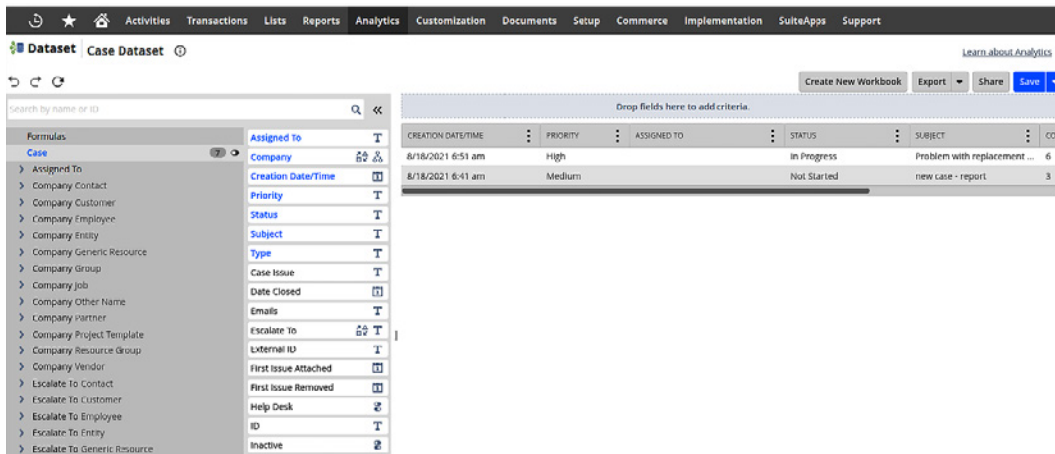


Figure 17.6 – The new dataset screen for Support Cases

We’re shown a list of field categories on the far left and then the files available within each category. This is also where we can apply custom filters via formulas to our dataset’s contents. The remainder of the screen is taken up with a preview of the columns that are currently selected for inclusion in the dataset. There are always a few columns preselected to get you started, but you can remove those and add your own choices as much as you need. Just remember that the dataset is just the collection of the data. The workbook is where you create the visualization for that data.

Since datasets give us access to all of the data in the account, we can include any custom fields and custom records in any dataset, and we can even combine datasets, with certain restrictions.

We do that with workbooks. You can create one or more visualizations based on the data in the dataset inside each workbook. For instance, if you build a dataset, as described earlier, with details from transactions but only for customers who have more than one credit card, you can then create two different reports from that data, and also a pivot table chart in a workbook. We can add a new table, a pivot, or a variety of charts to any workbook at any time.

For example, this is what the **Sales (Ordered)** template workbook, which comes with every account, looks like:

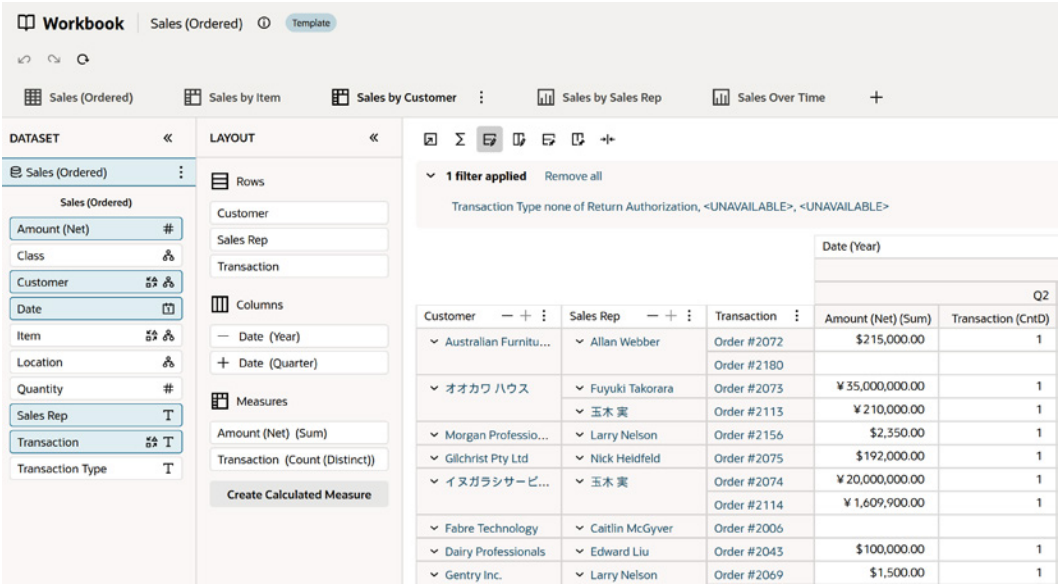


Figure 17.7 – The Sales (Ordered) workbook’s Sales By Customer pivot view

From this screenshot, we can see that a workbook can have multiple tabs, each with a different visualization of the data from the dataset. The author of the workbook sees this view, but they can publish it to other users who might only be allowed to view the results (without changing the workbook itself). This is done via the **Share** button, which allows the owner to control who else can see the workbook's data. These views can be placed on a role's dashboard, for instance, and the data can be exported to CSV files.

SuiteScripts can now also work with datasets and workbooks via the API, and this has become a way to deliver powerful integrations to other systems, again when saved searches fall short of a data analyst's needs. I'll talk more about this option in *Chapter 18*.

I've heard a lot of uncertainty from users about these more advanced features, by the way. They think they're too complicated to learn and use regularly. I've found that once you get the hang of datasets and workbooks, you can set up a quick search or a much more complicated view of the data with these tools, almost as quickly as we can create saved searches. Give it a try and see what you think. The NetSuite Help and SuiteAnswers have many useful pages on these topics, and quite a few videos to make learning about these features as easy as possible.

New in 2024.2 – Pivot Tables can now be exported directly to Excel

With this release, we have even more options for analysis and reporting from NetSuite, by making it easy to create pivots in a workbook and then send that data out to Excel for further work. For example, I recently created a workbook listing multi-shipping route details from a client's sales orders in a workbook and used a pivot table to report on that data in a way that helped them understand their shipping better. With this export to Excel option, they can now take that pivot and then share it among their non-NetSuite users and managers as they need. Check this feature out under the Help page titled *Workbook Pivot Tables*.

Linking datasets for even more flexibility

In the latest versions of NetSuite, we can now also link any two datasets, which gives us more flexibility when working with disparate sets of data. For instance, say you have a custom record listing warranties and a set of sales orders that those warranties are tied to in a *one-warranty-for-many-sales-orders* relationship. Natively, NetSuite won't let you choose fields from the warranty list for your sales order datasets, but you can create linked datasets instead. Create one for the sales orders and one for the warranties and then find a common field on both types that will allow NetSuite to link them for reports. When you create outputs for linked datasets, you can't create a table-style report, but you can create pivot tables or charts.

You might be wondering, what should we do when a client needs to routinely pull data out of the system, for use in a data warehouse or some other external application? In the next section, I'll cover my favorite option for meeting this requirement.

Creating exports with SuiteAnalytics Connect

When a client needs to access their NetSuite data from another system, we have a few natively supported options for making this happen. I will cover all of the integration options in *Chapter 19*, but when we’re talking about more than a little data, and the client needs to access that data regularly, we usually steer them toward **SuiteAnalytics Connect**. This is NetSuite’s built-in, easily enabled option that allows other applications to make industry-standard connections to NetSuite to pull data out of the system. This is a one-way street, if you will; we can’t use this feature to make any updates in NetSuite.

NetSuite currently has two different data sources we can use to connect to the SuiteAnalytics Connect service: `Netsuite.com` and `Netsuite2.com`. The second data source is newer and supports more granular permission controls, and so should generally be used with any new Connect development you create. The system will continue to support the older `Netsuite.com` option for some time, but it’s best to use the new option as soon as you can. Also, `Netsuite2.com` utilizes the **Records Catalog** browser for its table and field names, which is in line with all new SuiteAnalytics features, so again, that’s preferred.

This is what the SuiteAnalytics Connect setup screen looks like:

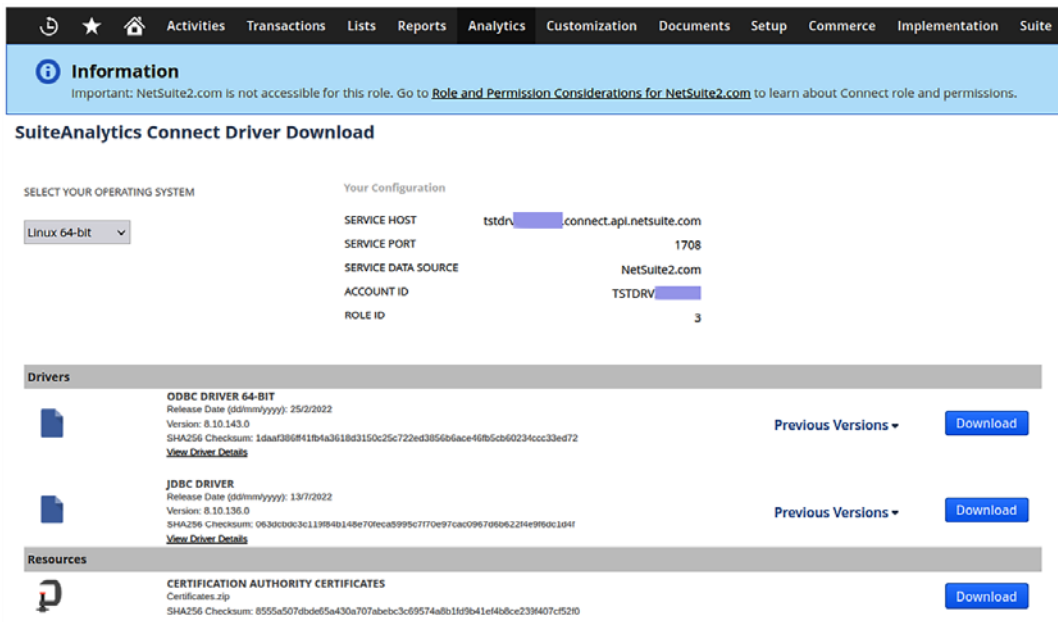


Figure 17.8 – The SuiteAnalytics Connect setup and driver download screen (for administrators only)

With this screen, an administrator may download the files needed to use a driver on another computer to make the connection to NetSuite and thereby execute queries for data.

Some people call this “ODBC” since that’s the most commonly selected option for making the connection to NetSuite at a technical level, but the system offers a couple of other standard options as well – JDBC and ADO.NET. Users running software outside of NetSuite use one of these technologies to make connections to NetSuite and then execute what are called **queries** to pull just the data they’re looking for. This is meant to be a programmatic connection, automated in some way by the client software, so this does require someone with technical skills to set it up. This usually makes the most sense when we are connecting NetSuite to a client’s data warehouse or business intelligence system. The technical users write queries that collect the latest sales data, for instance, and then they turn that raw data into reports or whatever they need in the external system.

These queries are written in an industry-standard syntax known as **SQL-92**. For people who have written queries with databases such as Oracle, MySQL, or Microsoft SQL Server, this should be very familiar. NetSuite publishes a list of the tables and fields that can be accessed via this interface (the Connect Schema), in Help on a set of pages called the *Connect Browser*. You can always find the link to this page in NetSuite Help. It lists every record type and field that’s available via this interface. Custom fields and custom record types are available here too; they’re just not listed in the browser (since they’re custom).

Let’s look at an example, just briefly, so that you can see the power of this feature. Say your client needs to export a set of details about their customers to an external reporting system. They want to see a list of all customers that have been created in the last 24 hours and get 5 details about each of them. The query for this might look something like this:

```
Select
  customer_extid,
  firstname + " " + lastname,
  date_first_sale,
  email,
  mobile_phone
From Customers
Where
  create_date >= '2020-01-01'
Order By
  lastname, firstname, date_first_sale
```

Quick tip: Enhance your coding experience with the **AI Code Explainer** and **Quick Copy** features. Open this book in the next-gen Packt Reader. Click the **Copy** button (1) to quickly copy code into your coding environment or click the **Explain** button (2) to get the AI assistant to explain a block of code to you.



The next-gen Packt Reader is included for free with the purchase of this book. Scan the QR code OR go to packtpub.com/unlock, then use the search bar to find this book by name. Double-check the edition shown to make sure you get the right one.



This Select query says we should focus on the Customers table and only include rows from that table where the record's creation date was after a specified date and then include a few named fields from that table. Queries can get much more complicated than this when we include joins to other tables. For instance, when we want to include transaction data in a query, we include the Transactions table and then join that to the Transaction_lines table to get the list of item lines from each transaction. **Connect Browser** spells all this out, so you can learn how to use it if you're the technical sort, and we can also use a query tool to explore the data we have access to via the so-called **system tables**. The oa_ table lists all tables we can see via queries, and the oa_columns table lists all the fields in each table. With these tools, we can put together just the right set of queries to pull all the data we need from NetSuite, at whatever frequency is right for the business. Just note that this is always a "pull" from outside of the system; there is no option here to have NetSuite "push" the data out on its own.

If you're interested, NetSuite supports queries such as the preceding one via other interfaces as well. Generally, they refer to this powerful feature as **SuiteQL**. It can be used to gather and export data from the system via SuiteAnalytics Connect, SuiteScripts, and also the SuiteTalk REST API. If you or your client has developers on the team, they should *really* check out these options as writing queries using standardized terms is very common in the IT industry. With SuiteQL, a developer can query the system from scripts running inside the account or via an integration from another server or service and then use that data as they need. And these queries can be much more complex than simple search criteria are usually. For example, a third-party integration with an e-commerce platform might make

it easier to gather a specific list of the items a client wants to sell by writing a query pulling together item and inventory and location data into one result set, versus trying to use multiple searches to do the same thing in NetSuite. All of the security rules that apply to the data inside NetSuite still apply when we're writing SuiteQL queries too.

I hope you take the time to familiarize yourself with SuiteAnalytics Connect and SuiteQL – they have proven to be lifesaving options in more than a few of my client implementations.

New in 2024 – NetSuite Analytics Warehouse

With Oracle and NetSuite working together more and more to offer really amazing new features, it's no surprise that the company now offers **NetSuite Analytics Warehouse (NSAW)**, a data store for not just your client's NetSuite data but any other data they want to add in as well. With NSAW, clients can combine data from their ERP and CRM records in NetSuite, with other data such as operations, warehouse, sales, and much more, and then report on the combined data with built-in BI tools or with other tools as they prefer. NSAW is an extension of Oracle's cloud services, but importing data from NetSuite and other sources is made as simple and yet secure as possible. Have your client talk to their NetSuite account manager for more on this subscription service or check out the documentation page titled *NetSuite Analytics Warehouse Overview* for more info.

Summary

In this chapter, I've tried to convey the main ways we can help clients gather their data, display it in meaningful ways, and share it with others. Searches and views are quick and simple to create for almost any user with a little training, but they can be so important to the flow of the client's business. Reports, workbooks, and datasets allow us to take a deeper look into the data and format it in standardized ways. To export the data, we put it in shareable forms such as PDFs and Excel files so that NetSuite's data isn't trapped in the system. Speaking of that, SuiteAnalytics Connect allows us to export data in a regular, industry-standard way to other systems.

Many clients rely on this to feed data into their data warehouse or another centralized reporting system. The freedom NetSuite clients have in working with their data should be clear from all of these options.

With that, we've finished talking about the native transactions and main features almost every client uses. In the next chapter, we'll dive in and learn how we can fill in the gaps in system functionality when they pop up with customization features such as SuiteFlows and SuiteScripts.

Self-assessment

For every consultant, you must think on your feet to solve client problems before they become major crises. Apply your thinking skills to these example problems based on this chapter's contents, and think about how you would solve them:

1. Your client has a search they've previously set up to help them find customers that meet a certain set of criteria. Now, they want to have the system send them an email whenever any results are found in this search so that they can fix issues and communicate with the customers. How can you set up a search in NetSuite that will meet this requirement?
2. The **Current Inventory Snapshot** report is very handy in that it easily shows the inventory levels for items in stock right now. However, your client notices that the **On Order** column pulls from both Purchase Orders and Transfer Orders. They want to be able to split those numbers into separate columns so that they can see just the Purchase Order and the Transfer Order numbers separately. How would you help them achieve this?
3. You've shown some users how to use the workbook feature to build a custom report, and they are getting into it. They want to know whether they can share one of their workbook's outputs with a group of users, via their NetSuite dashboards. There is a way to do this; what are the steps they need to take?
4. Your client, who sells software and services, is heavily relying on the **Advanced Revenue** features in the system, and they need to send some of their sales and revenue arrangement details to an external partner. They need to send some data immediately as it changes in NetSuite, and some data can be exported in batch mode. Which of the options discussed in this chapter could be used to meet these requirements, and which is best suited to this task?
5. When enterprise companies have smaller subsidiaries using NetSuite, they frequently need to send a lot of their data up to a parent company's servers so that it can be incorporated into the main business's reporting, and so on. A client is looking to use SuiteAnalytics Connect for this purpose from their account, but they're concerned about how well it will stand up to large volumes of data being transferred regularly. How can you help them get past their concerns and quickly establish whether this is the right solution for them?
6. A developer working for your client wants to collect sales data from NetSuite using SuiteQL. They say they're familiar with query writing already and have access to the NetSuite documentation listing the records and fields they're interested in. But as they test their integration, the system is reporting an error saying they don't have permission to access customer addresses. Without learning more about their technical work, how can you help them understand and resolve that permissions error from NetSuite?

Part 4:

Managing Gaps and Integrations

When a client's requirements can't be met by native NetSuite features, we turn to customizations. In this part, you'll learn about designing and implementing SuiteFlow and SuiteScript, plus how to integrate data from other systems, and how to migrate the client's legacy data into NetSuite – and how we go live!

This part of the book has the following chapters:

- *Chapter 18, Managing Gaps and Creating Custom Automations*
- *Chapter 19, Managing Integrations*
- *Chapter 20, Managing Data Migrations*
- *Appendix*

Managing Gaps and Creating Custom Automations

It is not terribly uncommon to find a requirement that cannot be solved by a native NetSuite feature or a partner's pre-built solution. When this happens, you must consider creating a custom SuiteFlow or SuiteScript.

In this chapter, we will learn how **workflows** and **scripts** can help solve problems and automate activities, and how best we can help our clients when it comes to gathering customization requirements and mapping out how they will solve them.

In this chapter, we will cover the following topics:

- Identifying gaps and needs for automation in requirements
- Customizing NetSuite processes with workflows
- Making NetSuite work for you with scripts
- Documenting customization requirements and solutions
- Managing customizations over time, including their deployments

By the end of this chapter, I hope you will be able to discuss these technical topics with your clients, having a good sense of the options that are available within the NetSuite customization platform.

Since workflows and scripts can touch upon almost any feature within NetSuite, you might be working with anyone from your client's company on these activities. Good non-technical communication skills will help you smooth out this process, and it always helps if at least one or more of the people you will work with have completed the basic **SuiteCloud** platform training.

Identifying gaps and needs for automation in requirements

Our clients can have many ideas for ways they wish to improve on the native NetSuite feature set, or ways they want to automate processes with more than one or two steps. Here are a few examples of requests I've heard over the years, so you're clear on what kinds of changes to the system we're discussing in this chapter:

- “Can you make it so sales orders (SOs) have a default location on the item lines, based on the customer?”
- “Our three largest suppliers need to send us their vendor bills in a custom file format, and none of them has an integration with NetSuite.”
- “Can you automate the generation of special journals to save us time processing our period closes?”
- “We need to create item fulfillments for SOs for accounting reasons, but we don't actually ship our products. Can you automate the creation of fulfillments to save our users a lot of extra work?”
- “We like the NetSuite Dunning module for notifying customers of past due balances, but we need to change some of its features.”

In the earlier chapters on gathering requirements, we talked a lot about how you cannot help a client solve their real-world issues without first really understanding their processes and people. For a consultant, that always involves being a good listener, asking the right questions, and suggesting the best solutions. Knowing the system well is a requirement (and yes, there is *a lot* to know!), but sometimes, you are going to run into situations where either no native feature does what your client needs or users do not want to perform a set of steps manually. As a reminder from previous chapters, your approach to solving problems like these should always flow in this order:

1. Look for native features. Try to convince the users to use them.
2. If possible, create a SuiteFlow (AKA a workflow) to streamline the work.
3. Look for partner solutions (also known as SuiteApps) that can meet the client's requirements or automate processes for them.
4. Create a SuiteScript as a last resort.

On that first point, sometimes users will say they cannot live with a native feature, but in many of those cases, they may not be clear on how small or large the ask is of their time, or they are forgetting how infrequently it will be a problem they need to solve. We try hard to quantify every automation request and to convince users that they will be OK with the manual process whenever we can. Of course, if someone says they need to do something 50 times per month, that is enough to make an automation worthwhile.

My recommendation here comes from the experience of working on many client projects where, for one reason or another, we could not complete all of a client's use cases with native features. SuiteFlows (or workflows) are automations we can create and maintain in the NetSuite UI, usually without any code or other highly technical underpinnings. Most intermediate or more advanced NetSuite users can learn how to create a workflow in an afternoon and then learn how to fine-tune and optimize them over time.

This is preferable to bringing in a third-party solution (a SuiteApp from the Marketplace) since there is no cost involved with SuiteFlows, other than someone's time to maintain them (if that's ever needed). But that option is better than having someone create SuiteScripts, in my opinion (and I have specialized in creating scripts for NetSuite clients for a long time). I am always concerned about not only how long these things take to create but also how difficult they will be to maintain in the long run and whether a customization might have a negative effect on the user's experience/system performance. That is why I recommend following the earlier steps 1 through 4.

The native features are all well documented in the system, and you can learn what you need to know about the available SuiteApps from their providers. In this chapter, we will dig into the custom workflow and script options. They are part of the SuiteCloud platform, as NetSuite calls their customization tools. The platform also includes things like **SuiteBuilder** and **SuiteTalk** (web services). Both workflows and scripts are very powerful, but as you will see next, they can also add complexity to the system.

Customizing NetSuite processes with workflows

Workflows in NetSuite are automations you can create via a point-and-click UI. They can work with almost any record in the system, and run either as records that are saved or edited or on a schedule in batch mode. So, you can, for instance, create a workflow that validates user inputs live while the user is editing an SO, or you can have a process that runs at midnight, finds all of your orders that require review, and flags them as such.

The most common use we have for workflows is to automate approval processes for records such as journal entries, purchase orders (POs), or vendor bills. Workflows are especially suited for this task because they support multi-step processes and can include branches in their process depending on variable inputs. For instance, we can create a PO approval workflow that allows us to send the transaction to different people depending on the value in a custom field or on the total of the PO. Workflows can loop back and reset the state of a transaction as well. For instance, if the second approver in a chain rejects a PO for some reason, they can send it back to the original creator or to the previous approver.

Workflows are best for handling edits to header fields on transactions, and less suited to making changes to sublist line details (such as an item line on an SO). Workflows do support this to a limited extent, but only for the **Items** sublist on transaction records. That means, for instance, that we cannot apply actions to the **Apply** sublist on a customer payment. However, for items on most other transactions, we can validate their values and then edit those line or body fields, create new records, send an email, or display an error from many types of transaction records.

The first screen we see when creating a new SuiteFlow allows us to first choose between creating a new workflow from scratch or starting from one of the NetSuite-provided templates. As of 2025, there are three templates available, and they are approval flows that can work with journals, POs, or SOs. It is not possible at this time for us to create our own templates.

When we opt to create a new workflow, we then choose the record type(s) the flow will monitor and a few other settings, such as how it will handle logging. Workflow logging appears on the record’s screen in the UI, so users can troubleshoot without having to look elsewhere. Notice, in the following screenshot, that unlike other places in the system, we do not get to select **Sales Order** as the record type; instead, we need to select **Transaction** generically, and then **Sales Order** as one of the so-called **SUB TYPES**. This allows us to apply one workflow across multiple types of transactions, which is a very common use case.

The screenshot displays the Oracle NetSuite 'New Workflow' configuration page. The top navigation bar includes 'Activities', 'Transactions', 'Lists', 'Reports', 'Analytics', 'Documents', 'Setup', 'Customization', and 'Commerce'. The left sidebar shows 'New Workflow' and 'From Template'. The main content area is titled 'Basic Information' and contains the following fields:

- RECORD TYPE ***: Transaction (dropdown)
- SUB TYPES ***: Sales Order, Statistical Journal Entry, Transfer Order, Vendor Bill (list)
- NAME ***: Validate SO Lines (text input)
- ID**: (empty text input)
- OWNER**: Peter Ries (dropdown)
- EXECUTE AS ADMIN**: (checkbox)
- RELEASE STATUS**: Released (dropdown)
- KEEP INSTANCE AND HISTORY**: Always (dropdown)
- ENABLE LOGGING**: (checkbox)

Figure 18.1 – Creating a new workflow, selecting a record type

Once you have made these selections, click **Save** to move on to define your workflow’s **states**, **transitions**, and **actions**. *States* are what the system calls the steps in the workflow’s process, and *transitions* connect one state to another. For instance, I have set up a workflow here that would start whenever a sales transaction is saved, review a few field values on the transaction, and if any fail that validation, then display a message to the user warning them of the problem. Otherwise, when all validations pass, the user will be allowed to save the transaction.

You can see in the following screenshot that we branch in one of two directions from the initial state in the flow. We place those conditions on the transition lines that connect each state box.

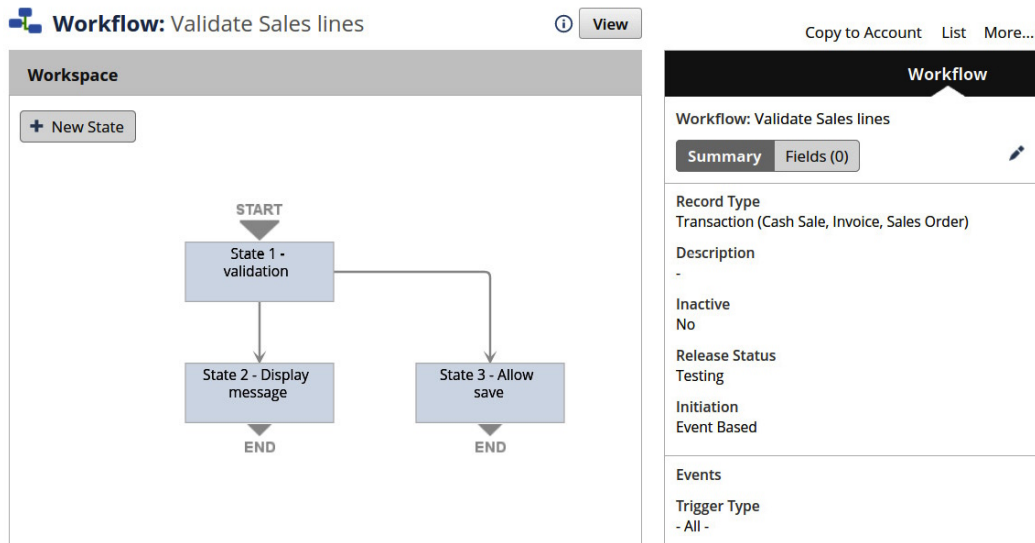


Figure 18.2 – A workflow with three states and two transitions

Inside each **State** box, we can enter *actions*, and those allow us to control the flow the user will experience while working on their transaction. There are quite a few types of actions, including adding a button to the page, sending the user to another page, locking the record from all further edits, setting field values, and displaying a message to the user (NetSuite Help lists all of the available actions: https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_4103695593.html). There's even an action type (called *Custom*) that can trigger a script whenever additional automation is needed. We can add as many actions as we need in any single state, and we can group them if we need to, as well. Each action we add within a state can have specific custom triggering events (like on the saving of a record, or on the changing of a field value), and we get to choose in which contexts the action should be processed (i.e., edit versus create, or user interface versus a web service update).

A special type of group is needed when we want to perform any actions on a sublist, and that is called a **Sublist Action Group**. Within one of these, we can perform a few actions on the **Items** sublist if there is one available on the transaction. So, for instance, if we want to look at a field value on an SO item line and then display a message to the user when it breaks a rule, we can do that with a **Message** action inside of a **Sublist Action Group**.

Workflows are very powerful. We can do a lot with them, well beyond just approval workflows, but know that they can get complicated pretty fast, and controlling their conditions, events, and so on is for more advanced users. Most of my clients have had at least a few people who could learn to do this in a short time, and so could create their own simple workflows whenever they needed to. However, once a workflow has more than a couple of states with lots of conditions and various events it is triggered on, there is definitely a skill to making it not interfere with other operations on the given record in the system.

Anyone can learn to do this, but practicing in a *Sandbox* or *Development* account is a good idea since we cannot ever afford to cause issues in a live *Production* account.

So, we like workflows because they are completely defined by a user working in a point-and-click UI in the system. But let us move on to learn about scripts next, which are developed in a different way while offering similar automation options.

Making NetSuite work for you with scripts

As I mentioned in the previous section, there may be requirements from a client that cannot be met via native features, workflows, or any existing SuiteApps. In these cases, we typically consider whether a SuiteScript can meet the client's needs next. A *script* is a lot like a workflow, except for a couple of differences:

- First, they are developed with what we call *code*, which is a set of script commands written out in a text file. NetSuite offers its own programming language for automating tasks within the system, and that is **SuiteScript**. It is based on **JavaScript**, a very common web programming language that has been around for a long time. There are a lot of developers around the world with JavaScript experience, although learning how to use SuiteScript is not nearly as common a skill.
- Second, scripts are executed by the system in the background, and they do not have any logs you can see when looking at a given record's screen. When a script changes a field's value, while we do see that on the **System Notes** on the record, we have to look specifically at the script in question to see its logs, or we can create a search whose results show the same data.

Scripts are great for the same sorts of automation steps we use workflows for – validating user inputs, automating the creation of related records, and other UI tasks – but they can also be used to create integrations with other systems and scheduled processes that run in the cloud on whatever frequency we choose. For example, we create scripts for clients to set field values on records so their users do not have to remember to do this, and to automatically create item fulfillment transactions from SOs when conditions indicate this can be done without a user having to click through those screens. We also create scripts to make web services call out to other systems, or to listen for web calls coming into NetSuite from other systems, for integration reasons.

In general, though, we do not ever want to get too carried away with all the powerful options scripts offer. We would not, for instance, want to build something completely unrelated to the NetSuite functionality on top of the system. As script projects get more and more complicated, they become difficult to manage and maintain, they can have a negative impact on the system's performance, and they can create confusion among users if they are not designed properly.

All of the SuiteApps in the Marketplace are solutions that combine SuiteBuilder custom objects (fields, records, lists, and so on) with scripts. However, these are all full-fledged products developed by professional developers with a lot of careful design time and plenty of testing applied to them. When you are considering developing a script for a client, always keep in mind that your time is limited, and so keep your automations as simple as you can to avoid the problems I mentioned earlier.

To create a script in NetSuite, follow these steps:

1. Develop the code file on your workstation and outside of NetSuite.
2. Upload that text file into the account, saving it to **File Cabinet**.
3. Create a new script using the UI.
4. Create at least one deployment for that script, so NetSuite knows where and when it will run.


The Help center has many great examples of what SuiteScript looks like, and it also goes into what it takes to develop a script. No fancy tools are needed, though. If you are just copying a code sample from a SuiteAnswer page, for instance, **Notepad** (free with Windows) or **TextMate** (on Macs) is fine for the basic task. You just need a place to copy and type the text that will not apply any unwanted formatting (so Microsoft Word and other similar WYSIWYG editors are out). The process is very simple – just open the text editor of your choice, paste some code in there (if you have it) or edit your own code, save the file with a `.js` extension, and upload that file into the `SuiteScripts` folder in NetSuite.

New in 2023, the NetSuite developers have published a repository of sample scripts for SuiteScript developers called `netsuite-suitescript-samples`. You can find it by searching on `GitHub.com` for this name or by visiting this URL: <https://github.com/oracle-samples/netsuite-suitecloud-samples>.

The following example is not a complete script, but it should be just enough code that you can see how we mix JavaScript and SuiteScript commands in a script to accomplish our goals:

```
function pageInit(ct) {  
  
    let status = ct.currentRecord.getValue('status');  
  
    if (status !== 'approved') {  
  
        ui_message.create({  
  
            title: "",
```

```
        message: "This PO needs to be approved, after you're  
            finished editing.",  
        type: message.Type.WARNING  
    });  
}
```

 **Quick tip:** Enhance your coding experience with the **AI Code Explainer** and **Quick Copy** features. Open this book in the next-gen Packt Reader. Click the **Copy** button (1) to quickly copy code into your coding environment, or click the **Explain** button (2) to get the AI assistant to explain a block of code to you.

```
function calculate(a, b) {  
    return {sum: a + b};  
};
```

Copy**Explain**

1

2

The next-gen Packt Reader is included for free with the purchase of this book. Scan the QR code OR go to packtpub.com/unlock, then use the search bar to find this book by name. Double-check the edition shown to make sure you get the right one.



The function in the preceding sample might run as a page is being loaded in the UI for a user, and when it is executed, it will check the current status of the transaction it is associated with. If that status is not **Approved** already, it will display a warning message to the user on their screen. There is some JavaScript in there (the `let` and `if` statements) and some SuiteScript (the `getValue()` API call), and they are used together to make a fully functioning script.

New script modules available in 2024/2025: LLM and PGP

With these latest releases, NetSuite has really stepped up its game for developers, incorporating the latest AI technology (in the LLM module) and adding a popular encryption option (PGP) to the Crypto module. With the LLM module, script developers can now utilize the power of AI built into the system in all sorts of new ways. Like any AI LLM service, dynamic prompts can be sent to the engine via a script, and a response will be returned to the script. The prompt can use a static preamble to prep the LLM, and the temperature and other settings the LLM recognizes can be controlled by the script. For instance, a script might be developed that summarizes a search's results, feeds those into the LLM, and asks for some analysis. Users can interact with the AI chatbot via these scripts. Clients can now also pre-define prompts they want to use more than once, such as a set of queries they will use to gather inventory or sales data. This is all really promising, and we expect to see great things created with these tools in the next few years.

The new PGP module enables a whole new way to encrypt messages and `https` request data using the OpenPGP protocol. Many banks and other organizations use this to secure the contents of data files being transferred over the internet, so developers should find a lot of uses for this new module in their scripts.

The system supports a variety of script types, and it is good to know how the most commonly used script types can affect a client's data or the users, so let us break that down next.

Client scripts

These scripts run in the user's web browser when they are working in the NetSuite UI. **Client scripts** can be triggered to start their processing when a record is first displayed, or when a field or list entry changes, and so on. We can validate their inputs while they are editing a record, for instance, or display a message (confirmation, warning, or error) when we need to. For example, we might create a client script if the business requires that we validate fields on an invoice line before they move to the next line. However, client scripts are more complicated to control and should be avoided for things such as creating new records unrelated to the one the script runs against or performing multiple complex searches. Instead, push those automations to the server-side scripts.

User Event scripts

These scripts run in the NetSuite server cloud, in the background. They can be triggered before a record is loaded by a user or any automation process before it is saved into the system, or immediately after. In my view, **User Event** scripts are the most commonly developed script type, since they have the lowest impact on the user's perception of system performance, and they are capable of making almost any change we need in an account.

However, the trade-off with this script type is that it does not run while the user is still editing a record – the user has to save the record before the script will complete its process. For example, we might create a user event script when the requirement says that every time an SO is created, we should send an email to a department manager. That doesn't have an effect on the user working in the NetSuite UI and can happen in the moments after the order is saved.

There are many reasons why we might create a user event script. We can use them to create new custom records to go along with a transaction after the user saves it. User event scripts can also be used to send data from an SO out to a third-party logistics system in something close to real time, when that is needed.

Scheduled and Map/Reduce scripts

When we have an automation task that does not need to run in real time against a single record, we can create one of these batch process scripts to run in the background instead. Both scheduled and map/reduce scripts run on a predefined schedule (or they can be triggered manually). However, **scheduled scripts** always run in a single thread, performing whatever logic you write into the code file and then exiting. **Map/reduce scripts**, though, can make updates to records in a multi-threaded way, using multiple *processors* or servers to do their work simultaneously. Either of these scripts is great for batch processing, running either at regular intervals throughout the day or once per day, week, and so on. We might create a scheduled or map/reduce script when we need to send a batch of statements to certain customers, or update all recently created work orders with custom segmentation, for instance.

Suitelets and RESTlets

When we need to create a custom page in the UI for users to visit to perform some custom function, we usually create a **Suitelet**. These are scripts that run when accessed via a menu option. They display whatever data we want on the page and allow users to interact with that data in a variety of ways. For instance, these pages very commonly display one or more tables of data, with a checkbox shown on each row, allowing a user to choose which one they want to act on.

Suitelets usually include a **Submit** button, which can trigger further automated actions. Just recently, I created a page like this for a client who wanted to see a list of POs that did not yet have a vendor prepayment applied. The user can select the orders they know should have payments applied, and when they select the **Submit** button, the script generates those vendor prepayments automatically, in no time at all.

A **RESTlet** sounds like it is similar to a Suitelet, but these scripts have their own separate use case. We create them when we need to provide a mini-web service in the account, always listening for requests to come to it from outside of NetSuite and performing some processing. This is usually some very limited action, such as updating SOs based on the data being passed in or searching for and returning a set of data in the script's response.

We do not use RESTlets when we need a full-blown integration with another system, such as when many record types need to be passed to it (since they have a relatively short timeout for sending a response), but they are great for a situation where just one kind of data needs to come into NetSuite or go out to another system in a limited way.

There are other script types as well, for instance, **portlet scripts**, which allow us to customize the contents of a portlet on user dashboards. They are all described on the Help page, along with all of the supported API calls, and more. However, this topic is very complex, so it is always recommended that users looking to create scripts take the NetSuite-offered training courses to gain a solid understanding before they go too far with development. And just as I said for workflows, developing in a sandbox or development account is critical to ensure we avoid issues in our live production account.

Once you have committed to creating one or more scripts, you should have an idea of how you are going to document how they work, which parts of the system they touch upon, and so on. Let us dig into that topic next.

Documenting customization requirements and solutions

When we work with scripts in clients' accounts, we usually need to create a couple of types of documentation to support our work. It is not enough to just have automations running in the account – users need to know how they affect their work, and administrators need to know how to maintain/change them in the future. The first document we typically provide to clients is a **requirements and design document**. This should be a very plain English document that any user can make sense of. This is not the place for a lot of technical terms or code samples or even what we sometimes call *pseudo-code*. Instead, I like these documents to have at least three sections:

- A clear statement of the problem we are trying to solve with customization
- A detailed functional description of the client's requirements for it
- A good, understandable description of the solution we will create to meet the business' requirements

Frequently, we should include a clear set of assumptions for things not specifically covered in the customization but that are still relevant to it. This is where we state what the automation will not do, for instance, or what steps users will still have to do manually.

With this, the client can review the document upfront and decide whether we really understand what they are looking to achieve. They can also use the document in the future to remind them of what the automation actually does (and does not do) in their account.

In one of these *functional designs*, then, when we describe the problem, requirements, and solution, we try to follow a few simple rules to make sure the document communicates well and nobody gets lost in the details.

Here are some guidelines you can follow when writing one of these functional documents:

- Problem statements are written from the user's perspective. So, *"Users need to be able to process a third-party payment from SOs"* is good. *"NetSuite needs to process third-party payments"* is wrong, since it is not the system that needs to solve this problem.
- Requirements should name the real features they will involve, should include quantifiable values, and should not mention how we will solve them. *"A script will automatically fulfill certain transfer orders"* is not good enough, since we do not know which transfer orders are affected, and also, no client ever asks for a script to be written. Instead, they ask for automation and let you tell them how it will get done. Better wording would be the following: *"Users require that when inventory is transferred from the secondary warehouse to the main warehouse, the transfer orders are fulfilled automatically."*
- Solutions should be specific but not technical. I do not like to mention the type of script I plan to create here unless I am sure everyone reading the document will be technically inclined (which is rarely the case). Instead, I focus on how the solution will work from the user's standpoint: *"An automation will run as transfer orders are saved into NetSuite and it will ..."* I do mention any custom configuration the solution will use, though, so custom fields, records, and lists should be named here. Lastly, I try to address how the automation will perform and how it will scale with the business – for example, *"The automation will add about 2 seconds to new SOs' processing time"* or *"The automation will be able to process up to 100 POs in any 60-minute window of operation."* This sets an expectation with the client that helps us avoid performance-related concerns later on, which can be a real problem if you have not nailed this down very early on in the script's development.
- If we create multiple scripts, and the client installs multiple other SuiteApps and solutions from other vendors, it gets very hard to keep track of what is a native feature and what is custom. When this happens, it is nice to provide an overview flow diagram as well, showing the client how their business processes have been customized and which parts are native features versus the custom things we have added on. This kind of diagram will be invaluable to the administrators, and we usually encourage clients to maintain the document after we are done working with them, as part of their long-term change management process.

Here is a very simplified version of a flow diagram, just to show you what these things can look like:

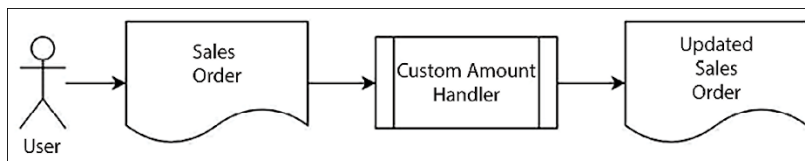


Figure 18.3 – A very simple sales process flow diagram

The main idea is that you need one of these diagrams for each business process that includes any customizations, and it should be very clear which steps use native functions and which use your customizations. It does not hurt to catalog your customizations in this same document, listing their names and the names of the records in the account that they read, create, modify, or delete. With this, the client's administrators can always tell where they need to focus their maintenance and upkeep over time, and this will also help them when they reach out to NetSuite or any other service for support after you have moved on from working with them.

Another aspect of workflow and script maintenance is code versioning and deployment planning, since customizations do not move themselves from one account to the next. I will talk about those topics next.

Managing customizations over time, including their deployments

Once you have added any custom things to an account, including custom records, searches, forms, roles, and potentially so much more, you need a way to manage those things outside of the system. This should be part of the client's change management process, and you will probably be asked to help them get that set up. If you have just created a couple of scripts and a workflow, it is not hard to help the client find those things in their account and to keep track of them in a simple spreadsheet or similar document.

The document you choose to use just has to list all of the custom things your solutions are using, including fields and searches, and the workflows or scripts they use too. That can be one large document listing everything or one customization-specific **deployment document** for each custom solution you have delivered. Handing this documentation to the client near the end of their implementation project can really help keep them organized and give them a sense that they know what they are going to live with.

But that approach starts to fail when you add a lot more components and have many separate solutions in use. At some point, maintaining the spreadsheet based on searches of object types from the system becomes too difficult. NetSuite does not let us "tag" each custom thing we create to a specific customization, for instance, so it can become confusing to figure out which solution a custom field is used with or whether it is still even needed in an account.

Beyond about 10 custom solutions, most clients will look for some help, since users will forget to report changes they make sometimes, and there are just so many things to keep track of. In these cases, we have recommended a third-party solution that can keep an eye on all custom objects created/modified/deleted in an account by any user, and then report on those changes in lots of useful ways. When this seems appropriate, I suggest my clients check out the **StrongPoint** solution in the SuiteApp Marketplace. I do not usually recommend any specific solution in cases like this, since I want the client to figure out for themselves whether someone else's custom application will work for their needs, but this is currently a solid suggestion to make.

In addition to managing the custom objects in the account, it is important for a client to know how to move those things from one account to another (such as if they begin using a new sandbox or development account). Moving SuiteApps is not a challenge, since they are packaged up and easily installed in any account we need. But moving a custom solution consisting of two scripts, three searches, and a few email templates from a sandbox takes a little more work.

For many years now, the main option we have had for doing this within NetSuite has been **SuiteBundler**. It is a point-and-click tool we can access directly in any account's UI to create so-called *bundles* of custom configuration objects we have created there. Those bundles can then be installed in any other account we have access to. This works great for creating new things in a destination account for the first time, such as when we first move a set of things from our development sandbox to our testing account. However, issues can arise when using SuiteBundler to update things that are already in the second account. It does not always handle overwriting or replacing things as we might want, and that can create problems if we are not validating the results carefully enough.

A newer script development option is called the **SuiteCloud Development Framework (SDF)**. This is a more technical tool; however, it is also much more powerful and, for technical users with a little training, it is a lifesaver. What it does is allow us to collect a set of code files and custom objects from one account (that might be your development or sandbox account) and then package that set of things into an **SDF Project** and deploy that into any other account when we are ready to. The main advantage of the SDF approach over bundles is that it gives us a great deal of control over deployments.

We can, for instance, grab a copy of a custom record type from one account, change its definition offline on our workstation, and then deploy that modified version to another account. This is possible because of the way the SDF defines a text file for each custom object we add to the project, and so we can edit that file when we need to.

In NetSuite 2023.1, we have a few options for integrating SDF projects into text editors on our workstations. NetSuite offers plugins for WebStorm from JetBrains and Visual Studio Code from Microsoft. With these editors (also known as IDEs) and the NetSuite plugin, developers can write and test their code from their workstations and then easily upload whatever they create into any NetSuite account they have access to. If a user does not want to use one of those editors, the system also offers a command-line interface tool, so users can perform the same interactions from their preferred terminal (in Windows, macOS, and Linux). With this option, development teams can automate their NetSuite deployments into their preferred **Continuous Integration/Continuous Delivery (CI/CD)** processes.

If you have never used this before, you can check out the Help and SuiteAnswers pages for more on this really helpful feature. There is also a simplified version of the SDF built into the UI for most types of custom objects, called **Copy To Account**. We can only use it to move one specific object to another account at a time, but many non-technical folks I work with have told me they love having this option available to them.

I hope you have seen how important it is to both track your customization work outside of the system (for disaster recovery and change management reasons) and also have a plan in mind for how you will move your custom work from one NetSuite account to another. The tools I suggested here are not terribly hard to learn, but they can be a real game-changer for larger clients who start to struggle with the complexity of all of their customizations.

Summary

NetSuite is a great product, combining so many disparate parts together into one streamlined system—CRM/ERP/web store, and so on. But since every business that uses the system is unique, it is not uncommon to find reasons why they need either customizations or automations to make their processes even more efficient. Workflows are great for approval and for many other processes where we just want a relatively simple condition applied to some data. They include branching flows in one of multiple directions, so we can really model the system's behaviors closely based on what they need.

However, when a workflow cannot do what is needed, we turn to SuiteApps from partners, and then to creating our own custom scripts. They add complexities, schedule delays, and usually also costs to an implementation project, so we keep scripts as a last resort in all cases. But sometimes there is no alternative. NetSuite does not charge extra to have a script running in an account, but there are always costs involved in their design, development, and testing to be mindful of.

When any implementation project includes a set of customizations, we need to help the client keep track of them in some useful way, which usually includes documents listing all of them and highlighting their functions. Managing those customizations' deployments from one account to another is yet another thing we must learn to do well, since no client has the expertise to do this themselves during a normally brief implementation time frame. You will do well to keep all these things in mind when the subject of customizing an account arises.

In the next chapter, I will discuss integrations with other systems and how we manage clients' requirements and expectations about them.

Self-assessment

Here are some automation-related questions you can consider. Think about how you would handle each of these situations if they came up in one of your NetSuite projects:

1. You are on a requirements gathering call with your client when they let you know that they do not like how customer payments must be manually applied to a set of open invoices. They ask if there is any way to have NetSuite apply the payment to the oldest open invoice first, and then the next oldest, and so on. How can you help them with this, without creating a workflow or a script?

2. During a working session with a group of sales reps where you covered sales commissions, Susan reminds you that they have a custom gross profit field in effect on sales orders. She asks if there is a way to change the commission calculation to use that custom gross profit field's value instead of the native fields. Is there a way you can accomplish what she is asking for with a SuiteFlow?
3. Your client's service business includes the use of charge records in their accounts, which will tie into the contracts they sign with their customers. They have asked you to automate an update on every charge record to make sure the data is prepared correctly before it gets applied to their Invoices. They will create approximately 10,000 charges per month (throughout the month) and generate something like 2,000 invoices (at the end of each month).
4. You are debating between these two types of scripts to automate this process: a user event that would run on the saving of every charge, or a map/reduce script, which would run once per day to process all of the new charges created that day. What are the advantages and disadvantages of each script type in this context?
5. *ABC Co.* is nearly ready to go live in its NetSuite account. Your team has created a set of scripts and workflows to customize and automate certain key business processes for them during their implementation. You have already provided them with a document listing each of those automations and their purpose. Now, they are asking if there is any way for you to include that information somewhere within their NetSuite account. There are a number of ways you could accomplish this, given some time. Can you think of a few ways you could help them with this request?
6. During the implementation project, your team has created many custom objects for a client. You have documented them all and are prepared to move them into production just before the scheduled go-live. Included in those custom objects is a set of forms that include many tweaks from the standard options. The client lets you know that they have already customized their forms in production, so they do not want you to deploy your forms, as your settings might overwrite their changes, and they are upset now. Think of a couple of ways you could use either SuiteBundler or the SDF to work around this issue.

Unlock this book's exclusive benefits now

Scan this QR code or go to packtpub.com/unlock, then search for this book by name.

Note: Keep your purchase invoice ready before you start.



Managing Integrations

For many NetSuite clients, the ERP system is the central hub for a collection of applications they use to run their business. Data needs to move securely and efficiently between each application for a business to function. Every consultant needs to know how to talk to clients about integrations, as well as how to see them through development and testing. If you're going to be more of a functional consultant, you might get a client started by talking about their integrations and then say, "...But I'm not the technical expert; for those details, we'll have you work with our technical team." In these cases, you still want to be familiar with these topics, so read on!

In this chapter, we will cover the following topics:

- Talking to clients about integrations
- Collecting information about and documenting integrations
- Managing data coming into NetSuite
- Creating custom exports from NetSuite

With these skills, you will be able to talk intelligently to clients about their integration needs and help them use the right solutions.

Just as in *Chapter 18*, when we're talking about integrations with clients, we need to understand the main features of the SuiteCloud platform, and any SuiteTalk and/or SuiteScript training will help the client follow along as you discuss development.

Talking to clients about integrations

We need integration whenever we need to bring data into NetSuite or send data out to another system. Integrations are very common in the NetSuite world, so every consultant helping a client through an implementation should be at least a little familiar with the topic. Gathering a client's requirements around any integration can be a challenge since the client doesn't usually have technical experience with integrations (most of my clients had someone else build the integrations they're using in their legacy ERP system), and we don't have experience with running their business. But so long as someone from the business can speak about the types of data they need to see coming into and going out of the system, and you bring your technical understanding to the conversation, it usually works out just fine. We, as consultants, need to always remember to speak without letting technical terms get in the way of the client's understanding, and we also need to be excellent listeners to pick up on every nuance of the requirements we're gathering (as always).

The following diagram shows a typical integration flow:

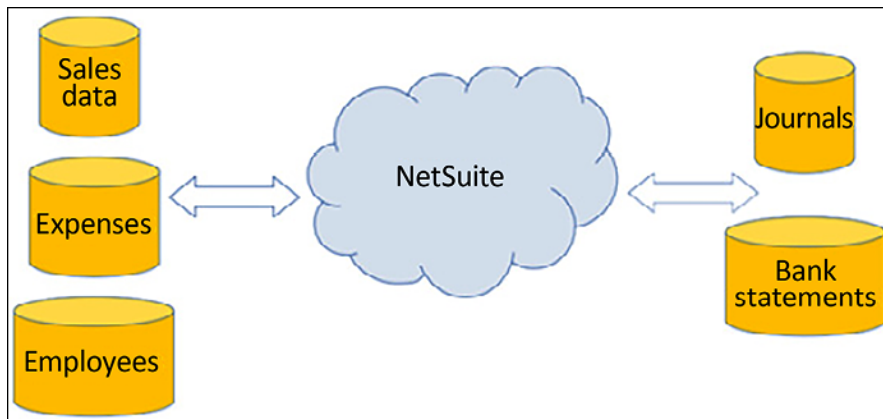


Figure 19.1 – A typical integration flow with data coming into and going out of NetSuite

Whether we're getting expense data from an external employee portal or integrating NetSuite's financial data into another ERP accounting system (as is often the case when a relatively small line of business in a very large enterprise uses NetSuite and sends their journals into a centralized server), I like to stick to terms everyone can understand. It's important to always be clear about what types of data we're focusing on and in which direction the data is moving. It's not uncommon for a new client to still refer to transaction names they used in their legacy ERP; when that happens, we can help them translate those terms into NetSuite record names. I avoid terms such as *upload* and *download* since they're relative, and just saying them doesn't tell you which direction the data is flowing in. I'll talk about *imports into NetSuite* or *exports from NetSuite* instead.

At the project management level, we should always call out any new integration requirements we learn of as they come up. If the integration requires development, and that might cause a delay in the rest of the project's schedule, the project managers and the client need to know that so that they can plan the custom development work accordingly. Once the entire team (your business and the client) is clear on what's involved, then new timelines can be put together. This can cause a change, for instance, in the go-live date, if the integration is critical and the work to develop and test it takes longer than the rest of the implementation.

Talking about data preparation for integrations

Every time we integrate with another system, we're moving data from one database to another, so we should always expect that a transformation of some kind must be performed on the data between the two systems. We frequently refer to this process as **Extract, Transform, Load**, or **ETL**. We extract data from another system or database, transform it for use in NetSuite, and then load it. This can be done just once in the context of the data migration needed for going live (which is the subject of *Chapter 20*), or on an ongoing basis within an integration.

For instance, when we bring data from a warehouse management system (WMS) into NetSuite, the WMS will have many pieces of data that will be useful in the ERP system, but for some, we won't have fields in which to store the data. An item fulfillment transaction might include details about the packages that were packed in the shipment. In cases like this, we might need to transform a date into a proper format or make sure we have a matching list of values for a select list. In NetSuite, we might find native fields to store the data in, or we might need to create new custom fields. The same situation can occur when data is heading from NetSuite to another application. The other application might need to be able to store some of our data in its system, in whatever its equivalent of a custom field is.

In some cases, we can talk to a client about using what we call a middleware service provider to meet their integration needs. These are companies that have built software systems that can take data from many sources and send it to any other destination. They support all the standard data formats, plus most have pre-built connectors for the most popular applications, such as NetSuite, Salesforce, Workday, and so on. If a client is already subscribed to one of these services, it's usually fairly easy to add flows to them to support new use cases. But otherwise, they can be expensive to sign up with and time-consuming to get started with. We should always keep this option in mind but also be mindful of the problems that it can add to a project.

I talk to my clients about *endpoints* (a connection between two computer systems, where NetSuite is one of those systems) when I'm trying to understand their integration needs. This gives us a simple framework where we can discuss all of the data that needs to come into NetSuite or be sent out from it. Once we understand the full list of endpoints, we can start to map out how each will be accomplished. I'll explain a few options to do that next.

Collecting information about and documenting integrations

Some NetSuite clients just need to integrate with one other application, for some relatively small part of their business. We always start to help these clients by searching the SuiteApp Marketplace to find out whether another business already has a package to accomplish this. With things such as taxes and warehouse integrations, as well as invoicing and payment handling, there are usually multiple options. But when we find that no app exists for their use case or the client needs to define and create integrations with multiple endpoints, we must take a couple of additional steps to make sure the work is planned and executed properly.

First, we must build a document listing all of the endpoints and their most important details. I usually do this in a spreadsheet since this allows me to easily build a table of information, and I can extend the listing any time I need. We always want to be able to refer to each endpoint by a common, easy-to-remember name, such as *Subscription Charges in NetSuite* or *Item Updates to the Data Warehouse*, so that's the first thing we include here. Doing this avoids a common issue I see at the beginning of these projects, where the client will say something like "All of NetSuite's data has to be sent to another system." That's not specific enough to work on, so breaking it down and naming each record involved is what's needed in these situations before we can start to make any headway. Having this list will also make it easier for everyone involved to stay focused later as we build out whatever is needed to deliver each integration from NetSuite's perspective.

For each endpoint, we like to list a few things in a summary fashion, based on what the client tells us is required:

- **Name:** Something that makes it clear what's transferred and, usually, its destination system.
- **Source system:** This is where the data is initially created.
- **Source data:** Which record type in NetSuite or the other system this endpoint is about.
- **Destination system:** Which system needs to receive the source data.
- **Destination data:** Which record type will store the received data.
- **Authentication method:** How the calling process will prove what the endpoint is to the other system. NetSuite supports a few industry-standard options, such as **OAuth** and **token-based authentication**.
- **Data transfer method:** How will we transfer the data into NetSuite or the other system? Common approaches include SFTP for file transfers or REST APIs for connections to web services.
- **Frequency:** *Real-time, hourly, daily*, and so on.
- **Estimated transaction volume:** How many records of the stated type are expected to be integrated each hour, day, and so on.
- **Remaining:** Whatever else is needed on a case-by-case basis.

Here's what that spreadsheet might look like, although you should apply your creativity to this task and make something that will work for you:

	B	C	D	E	F
1	Integration List				
2					
3	Integration Name	Sales data	Expenses	Journals	
4	Source System	NetSuite	Expensory mobile app	NetSuite	
5	Source Data	Sales Orders	Expenses	Journals	
6	Destination System	Data warehouse	NetSuite	ERP	
7	Destination Data	Sales	Expense reports	Financial	
8	Frequency	Daily	Hourly	TBD	
9	Estimated Volume	1,000 lines per day	50 per hour	TBD	
	Notes	Fixed Asset tracking			
10					
11					
12					
13					

Figure 19.2 – An example of a spreadsheet for integration tracking

For instance, if we know that some endpoints will use SuiteApps and that others will need custom solutions, we should include that detail in this document too. Getting a document like this started and fully populated can be a challenge on larger implementations, but it's worth doing. When we kick off this effort, I explain to the client that the document will be modified over time and won't be finished until they're live on NetSuite.

Once you have started that list, it's useful to create an official **integration long-term planning** guide. This is typically done in the form of a Word document, with sections and details we think are relevant to all of the client's integration needs – whether they will be processed manually, whether we're creating automation, or whether the client will use an off-the-shelf solution. This gives them a single place that the business's users can refer to later to answer questions such as *Where do our Sales Orders come from?* or *How often should our customer records be synced with Salesforce?*

We also include other details in these documents, including topics such as integration security issues and how the integration work will tie in with the implementation's data migration plans or any related customizations being developed. This is also where we identify the ID fields used by both endpoints. When we're storing another system's IDs in NetSuite, we frequently use the **External ID** field, available on each native and custom record.

It's a good idea to incorporate any special data handling routines into this document. We try to nail down things such as date or phone number formatting in one place and explain how lists such as `Items` and `Accounting Periods` within any of the integrations will be handled. For instance, if a banking integration requires NetSuite to send payment files to a secure remote server, the high-level details of that connection can be described in this document. Lastly, for each integration endpoint, it's a good idea to make it clear which system is thought of as the *system of record*; new data will be created in that system, whereas the other system just inherits those records once they're created.

By having the integration listing and the overall architecture documented as much as possible early on, we can launch into the work of defining the new scope in a contract with the client (if that's needed) and creating whatever deliverables are called for.

In the next two sections, we'll break this work down into two separate streams – imports and exports from the system.

Managing data coming into NetSuite

When we talk about integrating data into NetSuite, there are several things to decide on as we initially plan the work:

- Will NetSuite pull the data from another system, or will the other system push it into NetSuite?
- What are the technical options for accomplishing this integration?
- Does the integration need to be automated, or can a user perform a CSV import to bring the data into NetSuite instead?
- What actions are needed on the NetSuite side? (In other words, will we be creating, modifying, or deleting records?)
- Who will create the integration that performs these actions?

In some cases, these things are easy to map out – the other system has a connector already, and we just install and configure it in NetSuite, and we're done. In other cases, though, this can be tricky. When we need to build some custom integration in NetSuite, the first topic that comes up is usually one regarding the *scope* of our professional services contract.

It's common in integrations like this for the scope to be called out in our initial **statement of work**, for instance, but many times it isn't. We might need to negotiate with the client to add services work (at some new cost) to the contract before building something new. But this depends on your delivery business model – how the initial contract was set up. If you're working on an open-ended time and materials contract already, you can get approval for the work and then go for it.

Getting two systems to talk to each other can occasionally be time-consuming and difficult. As I mentioned previously, when we integrate another system's data into NetSuite, the two separate databases might not have a lot in common, and that usually means we need to find a way to both transform the data before it's transferred and also create custom records or fields in which to store it, in the destination system (NetSuite or another application). Once, I worked with a client who wanted to bring their legal caseload data into NetSuite, to make it available on the invoices they send to their customers. We had to integrate that data from the system their legal users worked in, store it in a custom record, and also tie each of those to an existing invoice in the system to support their need for a link between the two records.

In some cases, when data comes into NetSuite, we might want to create a custom record type for this purpose, allowing us to define a place where all of the data will reside within NetSuite, without adding a lot of unusual fields to a native record type. Then, we can leave the data there and include it in reports or whatever we need, or we can incorporate some of those fields into native records as needed. For instance, if a business uses a third-party system to track most of its day-to-day service operations work, it might want to import that data into a custom record called something like *Daily Services*. Then, that data can be synced to invoice lines or just reported on to make it useful within the ERP system.

As I mentioned earlier in this section, we always want to find out how frequently data will need to be brought into NetSuite. When the frequency is low, such as once per month, we will suggest that a user perform a CSV import rather than create an automation. This saves on development, and users can usually learn this approach in just a single training session. When the frequency warrants it, though, we have a few other options, where some type of automation running in NetSuite can be used to import data for users. We'll look at this in the next few sections.

Option 1 – SuiteScripts

The most common option in these situations is to create a SuiteScript of the **User Event** or **Map/Reduce** type. Depending on the need, we can create a script that runs in real time (if the triggering event happens in NetSuite) or on a predefined schedule. Either way, these scripts can be made to reach out to an external system, such as a web service or a **secure file transfer (SFTP)** server, gather the intended data, and integrate it into NetSuite wherever it belongs.

This is technically possible but requires a good deal of work. See *Chapter 18* for more on SuiteScripts in this book, and the NetSuite Help section has good starter info as well, along with examples you can use for things such as integrations.

Option 2 – SuiteTalk SOAP

When the owners of an external system are OK with pushing their data into NetSuite – that is, they can develop the integration themselves – then we talk about the two **SuiteTalk** web services options that are currently supported. **SuiteTalk SOAP** is the more mature option, having been available for many years. It's based on an older XML-formatted technology, but it's fully featured and is used by many, many integrations already. The SOAP standard is used by all sorts of applications (not just within NetSuite), so many developers will find it easy enough to begin importing or exporting data with NetSuite using easily accessed and configured development tools. NetSuite publishes a sort of data dictionary called the *Web Services Description* for SuiteTalk SOAP that gets updated twice a year, along with the rest of the product. Search in NetSuite Help or SuiteAnswers for the *SuiteTalk* section to learn more about this topic and the next, since they're both SuiteTalk web services options.

Option 3 – SuiteTalk REST

SuiteTalk REST is a newer option, and as of 2025, it's a generally available feature. About 150 native record types are now **generally available** (that is, **GA**), such as Customers and Sales Orders (SOs), which means NetSuite fully supports their use with the REST API; the API's details won't change without an announcement in advance. That makes it a great choice for NetSuite clients to build their integrations upon. REST is a standard that most web developers will be familiar with since it's become the most popular communication method among modern web applications and services.

With either the SuiteTalk SOAP or REST option, though, someone outside NetSuite has to develop and test an integration from an external server. This is usually a challenge since those people probably won't have NetSuite experience or even access to an account, other than the client's. You can help them with this by providing them with answers to questions about NetSuite and the APIs, ensuring they complete their tasks on time.

Option 4 – Middleware

As mentioned, another popular method for integrating NetSuite and other systems is the middleware option. With this approach, systems integrators contract with a third party, usually another cloud SaaS system, to act as the intermediary between NetSuite and wherever the data is actually coming from. There are a few popular companies with offerings in this space, including Oracle's own **Oracle Integration Cloud (OIC)**, Boomi, Celigo, and Jitterbit, but there are many services to consider in this market, each with its advantages and disadvantages depending on the client's use cases. One thing they all have in common is that they require a monthly or annual subscription fee, so this has to be factored into a decision to go this route.

Option 5 – SuiteScript RESTlet

When another party is OK with developing an integration, but the data they want to send cannot come into NetSuite through SuiteTalk (which is somewhat rare), then we can create another type of script known as a **RESTlet**. I described them in *Chapter 18*. These scripts are like mini web services running in NetSuite. They *listen*, as we say, for another system or service to call them, with either data in the request or a request for us to send them NetSuite data and respond accordingly. We develop them to use a custom API (of sorts) with the external party. In the case of data coming into NetSuite, the other party might be an external e-commerce web store that wants to send data to NetSuite, and then we might create a RESTlet that receives those requests and creates SOs in NetSuite using the data. We usually choose this option when the third party already has an API specification and they need NetSuite to work with that, versus adapting their system to the NetSuite APIs. You can find good examples of RESTlets in the Help or SuiteAnswers section.

NetSuite Connector – new and improved in 2025

In 2024, NetSuite acquired software previously known as FarApp and renamed it as **NetSuite Connector**. This fantastic new integration tool makes it easy to set up *point-to-point* integrations between NetSuite and a set of other services, including e-commerce, **point of sale (POS)**, logistics, and others. With NetSuite Connector, clients can bring import data into NetSuite from systems such as Amazon.com, Shopify, and so on, or export data to them, with minimal configuration and no additional customization. This is going to help a lot of clients streamline and simplify their order and fulfillment integrations. The full list of external systems Connector supports can be found in the NetSuite Help on the page titled *NetSuite Connector Supported Storefronts and 3PLs*.

Next, let's go over the options a client has when they need to send data from NetSuite to another system.

Creating custom exports from NetSuite

When a client needs to export data from one system to any other system, we usually push for the simplest options we can find and only resort to building something custom when those tools just don't fit the bill. As described in *Chapter 17*, our first choice for exports should always be **SuiteAnalytics Connect**. As a reminder, this is the option that allows a client to connect to NetSuite with one of three industry-standard methods (ODBC, JDBC, or ADO.NET).

They then execute *queries* to collect any sets of data they need from the system. The client's technical people can create automation that runs outside NetSuite in a variety of databases, data warehouses, or applications to pull data out on whatever schedule they need. The NetSuite Help page explains how to use the Connect feature, and it's very popular for this use case. Most developers already know how to use Connect's SQL queries, so this is usually something they can do themselves, once we help them get started and get their tools connected to the account.

When Connect doesn't work (not very often), we can fall back to something custom running in NetSuite for exports. We start by suggesting that the users run a search and select one of the export options from the results screen (CSV, Excel, and PDF options are always available), or sometimes we will set up a saved search that runs on a schedule and sends its results to an email address. But when these options are rejected, we have the option to create a script.

For instance, say a customer needs to have certain details captured from every new (and approved) purchase order, and those details must be sent to some other system for additional processing. We could do that with a user event script running to save the record in NetSuite if we need to export data in near-real time. This is not ideal since we would need to include additional logic in the script to handle any errors that occur, which will depend largely on how critical this operation is to the business and the fact that this kind of process can slow down the system from the user's perspective.

Something more common for exports would be some sort of batch process. A scheduled or map/reduce script can be created to gather a set of records that have been created in the system at the end of every day, for example, and send them to another system, in the form of a file or a stream of data sent to a web service. This is also a common option we select when the data needs to be sent out of NetSuite using another company's proprietary API. A script is written in that case to format the data however the third party requires it and sends it in a way they can accept.

New script features available in 2025

New in the latest release, NetSuite has added more features to its N/LLM module, allowing clients to develop powerful AI-enabled features, driving more business process efficiencies via automation. Along with the new native **prompt records**, clients can pre-define a list of the prompts they want to run through the LLM module and so drive consistency into their processes (and those prompts can be generated/updated via scripts as well!). The LLM module now also supports image processing, allowing scripts to upload an image along with their prompt and then get info about it in the response from the LLM. Responses can now also be *streamed*, allowing users to see the response as the AI tool generates it, for a more interactive session. Check out the 2025.1 Release Notes for more info on these exciting new features.

For all of the export options mentioned in this section, keep in mind that scripts are tricky to develop and get running correctly, and they add costs and delays to the schedule, so we want to reserve this option as the last one we'll consider for most client implementations. However, they can be really powerful, and if the frequency and data volume warrant the work, they can be well worth the effort.

Summary

Successfully running a business in the 21st century almost always requires integrations between systems. We like to think that NetSuite eliminates the need for a lot of integrations that used to be required, but that doesn't mean NetSuite can operate in total isolation. We want to help our clients choose the best option they can find for every business process, although sometimes that means someone will need to make two or more systems share data.

In this chapter, we learned how to assist the client in the planning stages for their integrations by helping them understand each integration endpoint that their business requires. Once we've done that, we can help them choose the right technologies to meet each need.

Data can be imported into NetSuite by customizations running in the account via middleware solutions, or by third parties developing custom solutions based on NetSuite's APIs. We can export data from the system via searches, CSV exports, SuiteAnalytics Connect, middleware solutions, or scripts, depending on all the variables in each situation. No matter which option we choose, we must plan for the complexities involved in any custom development work to keep our implementation projects on track and schedule.

In the next chapter, *Managing Data Migrations*, we'll talk about bringing the client's legacy data into the system and how this is not something we wait until the last minute to address. Migrations need to be planned from the beginning of the project to make sure they're just as successful throughout the rest of it.

Self-assessment

Here are a few more practice questions for you to consider. As in the previous chapters, think about how you would respond in each situation:

1. Your client is about 1 month away from going live when they let you know that they need help getting data from their expense tracking system into NetSuite on an ongoing basis (not just for cut-over). In what order should you research the options available to make this integration happen?
2. You've started to build a list of the integration endpoints your very large client needs with NetSuite in a spreadsheet. They mention that one particular integration will need to transfer something like 1 million records from their legacy accounting system into NetSuite. What sorts of questions should you ask at this stage to make sure you understand and address this requirement properly?
3. Just as you begin working with a new client, you learn that they will only have five accountants working in the system; there will be no other NetSuite users. They intend to integrate all of the data that those people will work with from another system – that is, the application everyone else in the company works in every day. How should you adjust your consulting to support this situation?

4. A client is thinking about how to export their very high-volume daily sales transaction data to another system. They want to know whether the SuiteAnalytics Connect feature will be able to perform their transfers as quickly as they need. They say that they might need to send around 10,000 orders out of the system every day, and each order might have 100 lines or more. They need to know that this data can be synchronized from NetSuite once every 15 minutes. How can you help them set up enough test data in NetSuite to run a few timed tests so that they can answer this question?
5. You've been planning to use a middleware partner to transfer journal entries from NetSuite to another system with your client, at which point they tell you that their contracted developer is not going to be able to complete the work. They ask how you can help them resolve this issue. Assuming you and your company lack the technical expertise to finish the work yourselves, how else might you assist them in completing this task? (What other resources can you call upon, for instance?)

Managing Data Migrations

Since most clients have some kind of system they were using before NetSuite, successful data migration is almost always required before they can go live. Migrations do not happen right at the end of the project, though. You and your clients must carefully plan for a successful migration, and that plan needs to include test-run imports long before the exciting go-live date arrives.

In this chapter, we will cover the following topics:

- Planning for the data migration process
- Performing imports for data migration testing and other reasons
- Planning the cut-over activities
- Migrating inventory and trial balances
- Performing the final data imports before go-live

After completing this chapter, you should be able to talk to a client about their data migrations and help them plan those. Most clients cannot do this on their own, so experienced guidance is always necessary to get through this milestone.

When planning, testing, and executing a data migration, both functional and technical consultants need to work with the various owners of each type of data the client will use in the account – items, lists, entities, and transactions (plus any custom records the client will go live with). The clients should be familiar with the NetSuite basics and also with using the CSV import feature, since that is the most popular way we bring data into an account.

Planning for the data migration process

When is the right time to start planning for the data migration process? The answer is soon after the implementation project begins. At this stage, you should know how the client has been running their business before they came to NetSuite, and so what types of data are needed. Ideally, in your company's sales process, you should have already addressed the types of data that need to be migrated, how much historical transaction data will be required, and which open transactions should be included in this process. The most common approach we see with clients coming into NetSuite for the first time is to migrate these data points:

- Lists (Chart of Accounts (COA), locations, subsidiaries, departments, and so on)
- Items (inventory, non-inventory, assembly, service, and so on)
- Entities (vendors, customers, partners, and so on)
- Historical transactions for a short period (6 or 12 months at most)
- Open transactions (sales orders, purchase orders, and bills)

Anything the client asks for beyond these basics should be carefully discussed and reviewed since, for instance, trying to bring in all of the company's historical transactions in detail can be a huge job that can massively slow down the project's timeline and involve a lot of extra effort for all the parties involved. In the past, when clients have insisted on this or something such as five years of transactions, we have made sure they understood how difficult this can be and how it would cost them extra, since this is not usually specified in their initial contract. Most clients will understand limiting their migration to data they can use in NetSuite, and that helps avoid a long, drawn-out migration process.

One exception to this process, which I have only had to make once, is when a company is moving from one NetSuite account to another. In this case, such as when a subsidiary is split off from their parent company, the demand to bring all of their NetSuite data into the new account has more weight behind it. NetSuite does not have an easy option for doing this, though, so planning for this kind of migration is even more important in this situation.

The plan for data migration usually consists of three phases once the schedule has been planned: *data preparation*, *testing*, and the *final migration*. To make a schedule, we have to start by working out the dates for the plan's milestones as early in the project as possible. After that, in each project's migration plan, the next thing that must be clear is who is responsible for the data imports.

It should be stated in the client's contract whether they are expecting the consultants to do this work or whether they should do it themselves. This choice will drastically affect the work you have to do. In general, with NetSuite-led implementations, we ask the client to take ownership of this task, and we guide them to a successful outcome. In cases where we are contracted to deliver the migration ourselves, we have teams who specialize in this activity, and they work with the client's users to gather the necessary data, clean it up, and run the test and final imports according to a schedule that has been laid out from the beginning.

In all cases, the implementation team should be there to guide the client on this topic. For smaller companies, we can usually just hold a meeting with the client's stakeholders once, explain the schedule of activities, and let whoever is going to run the import begin their work. Most small companies can manage this on their own, with us making sure they stay on schedule and perform the tests they need to. Nobody wants to find out two days before go-live that all of the customer records include incorrect values or that the phone numbers are in a format NetSuite will not work with.

For larger companies, or projects where we have a lot of data to migrate, we typically want to write up a **strategy document**, spelling out all of the data migration details, including the data to be imported, the schedule we will follow, notes on any special data handling or cleanup we know will be needed, and so on. We do not consider this a contract, but rather a guide everyone can follow once it has been reviewed as we work through the migration together.

It is at this time that we want to make sure we all understand exactly where all the data to be imported originates from. In many cases, it does not all come from just one legacy system, but a set of applications. For each of those, we want to be sure we know how the data will be extracted and in what format it will be in at that stage. For instance, it is not uncommon for a client to say that their legacy ERP system makes it difficult to extract all of the data we need or to get it into a format more or less like NetSuite expects. In these cases, we need to plan to perform the extraction and *clean up* the data, making it ready to be imported into NetSuite. Some clients need help with this step, but many have people on staff already who can transform data files from one format into the NetSuite CSV format pretty easily, with pointers from you. Some contractors can help with a process like this as well, of course.

Note: How we handle IDs

NetSuite will assign its own “internal ID” to each record created during an import, so you don’t have to worry about that. But often, records come with some sort of ID from an external system. We usually map those IDs to the **External ID** on the record, which is a native field available on pretty much every record type in NetSuite. That’s great, especially if you need to keep a record in NetSuite in sync with some other system. The external ID can be referenced later, for CSV-imported updates, for instance. The only issue with using that field is that NetSuite doesn’t expose it in the UI, so sometimes you might want to create a custom field that’s shown to users and sources its value from the external ID.

That data cleanup can involve all sorts of changes to the data. The COA is usually among the first things we migrate, and we can make small tweaks to the list during this migration as well. But generally, we don’t agree to major COA changes at this stage. Addresses, dates, email addresses, and phone numbers need to be in a format NetSuite will recognize.

Fields that use a list of values, like `State` or `Country`, need to have the list defined in NetSuite first before the records can be imported. Sometimes we create new custom record types or custom transactions to store data from another system, depending on how the client wants to use it in the system.

For more tips on data migrations, check out this NetSuite blog post:

```
https://www.netsuite.com/portal/resource/articles/erp/erp-data-migration.shtml
```

Once you have a plan and you know where all the data is coming from, you can start to see whether it is ready to bring into NetSuite. That is why testing is so important, so let’s dig into how we can do that next.

Performing imports for data migration testing and other reasons

Once you start to collect a set of files from another system, you want to bring them in, in a limited fashion, to test the waters for the migration. The first step in this process is typically helping the client know what NetSuite is looking for, with each of the lists and record types they want to import. The NetSuite services team typically provides templates, in a set of Excel files, to show the client which columns are to be included with each import, which of those are required or optional, and what the data type for each field should be.

The following table shows what they usually look like, including extra rows explaining how the data should be formatted, and so on:

	A	B	C	D	E	F	G	H	I	J	K
1											
2		Field Name	Name	Address 1	Address 2	City	State	Country	Zip Code	Phone	Email
3		Data Type	Text	Text	Text	Text	List	List	Text	Text	Email
4		Length	60	80	80	80	2	50	10	20	30
5		List values					States	Countries			
6											
		Notes					Format: XY		Format: 999-999-9999	Format: xxxxxxxxxx@xxxxxx.yyy	
7		Required?	Yes								Yes
8		Field ID	entity	address1	address2	city	state	country	postalcode	phone1	email
9											
10											
11											
12											
13											

Figure 20.1 – Template for an entity record

Sending a client a set of templates and guiding them in their use is typically enough to get them started. They will determine which columns in the data files from their legacy system need to be tweaked to match the NetSuite standard, as well as where the lists for things such as customer category or the addresses that have been assigned to each vendor will need to be adjusted to work in the new system.

Be prepared, however, to provide some clients with additional assistance; this can be tricky, as most companies do not assign people with a lot of data-handling experience to this task.

If you need to help a client create a CSV import template for a record type that is non-standard or one you just do not have a template for yet, here are the steps I follow to make this as painless as possible:

1. Go to the record in NetSuite and create a view showing all of its fields.
2. Create at least one record if you need to.
3. Use the CSV export feature to export the data of that view into a CSV file.
4. Open that file in your favorite editor (a spreadsheet program or a text editor with special CSV-handling features).
5. Add notes and additional details to the file to help the client understand what they need to provide for this record type.

While a client is preparing their CSV import files, some conversion of the data is almost always necessary. For instance, phone numbers and dates may not be in the correct format, or list values for things such as GL accounts and vendors may not be exactly the same between the two systems. Someone will need to carefully think through what the data in NetSuite should consist of in each case and make sure these rules are applied to all of the data being migrated into the system. This can be time-consuming, especially for those without any experience, so consultants should always look to offer tips and tricks to streamline and speed up the process.

Once the client gets used to working with your templates and they have someone convert their files into the NetSuite CSV format, they are ready to start running test imports. The first choice in that process is which account they should import into. If they just have their production account, then that choice is easy; be sure you plan with them to remove any test imports from that account, which we usually do as part of the cut-over activity. If they have one or more sandbox accounts, use the one you think of as the development account for your first import tests.

With this approach, you can work out any issues in the source data or the import mappings. Once you have successfully tested every record type in the development account, for some percentage of the total dataset, you will know that the migration into the next account – typically a user acceptance testing (UAT) sandbox – should go smoothly. In that scenario, we typically perform a full set of migration tests into the UAT account for all of the data we have at that time. This helps us ferret out any potential issues with bad data in the larger set of files, and it also helps the data migration team adjust to any changes the rest of the implementation team might have made.

For instance, if the people working on sales orders decided to add a custom field to that record, and they make it mandatory, the data migration team needs to know that and ensure that they are creating sales orders correctly before going live.

Next, we will check out the data testing that occurs in these phases.

Testing the data in phases

We typically do tests for migrations in stages; that is, we recommend starting small, making sure that a few records have been imported correctly. With that, we will have more confidence that we can import 100 or 1,000 records of the same type. We pause here and validate that data as best we can. It is very important that we take notes of any errors that occur during these test imports; they will tell us what additional steps need to be taken on the source data files as preparation for all future imports.

Data migration performance

One other thing we are testing for at this stage is how long each import takes. If the client is bringing something like 100,000 transactions into the account, for instance, we need to know how much time each import requires so that we know when to start the final migration. In some cases, for older historical transactions, we can import those into the production account in advance. But in most cases, since the cut-over data is very time-based, we do not want to capture the source data until just before the go-live date. It is in these cases that knowing how much time we need to reserve for the final migration is so important. If the import takes a week to complete, we cannot start two days before go-live. This is usually not an issue, especially in cases where we have stuck to the records and data volumes I mentioned earlier, but as always, planning and knowing how the imports will perform is the key to success.

It usually does not take long at all to get the basic lists – the COA, subsidiaries, and so on – imported. Those types of things can usually be imported well in advance of the go-live. Then, we move on to importing items. This can be more of a challenge, depending on the source data, but there is always a solution to any problems that might arise. We move on from items to the entities, working with each team of data owners, either to assist them in prepping or importing their files or doing the work for them, if that is what is called for.

For very large datasets, we have sometimes had to bring in an integration to help with this data migration process. Tools specifically made for this purpose will be useful if you have hundreds of thousands or millions of transactions to import and a very short time in which to complete that step. You can search online for tools like this generally, but NetSuite partners such as **Oracle Integration Cloud (OIC)**, **Boomi**, and **Celigo** specialize in this sort of thing.

They can utilize NetSuite's multi-threading options to create multiple records at the same time, assuming the account has already been set by NetSuite to allow that. This option requires careful planning and testing, though, to ensure everything will work as expected in the final migration. If your client's account isn't performing at the speed you need for CSV imports or integrations, you can work with the NetSuite account manager to talk about things such as higher server tiers and one or more SuiteCloud Plus licenses. SuiteAnswers has a nice page on how these things affect performance, named *NetSuite Service Tier Structure*, which is great to review before you get involved in an integration requiring them. Or refer to the Help page titled *NetSuite Service Tier Structure*.

Good communication from all parties involved is key here, to make sure we do not let little problems turn into bigger issues by not addressing them quickly. It is also key to making sure everyone is clear on the roles and responsibilities each person on the team has in this overall process. I have seen it happen more than once that a project has been derailed when both the consultants and the client are pointing fingers at each other and saying, “I thought you would do that.” In other words, it is always OK to be extra explicit and clear in all emails, phone conversations, and meetings about who is doing what and by when they are expected to complete each task. As a consultant, you are expected to avoid embarrassing situations like this.

Once you have performed imports for each record type, including at least something such as 50% of the total dataset for each type, you can move on to planning for the go-live.

Planning the cut-over activities

Data migration is typically the most important thing that happens just before going live, but it is not the only thing your team of consultants and the client’s team will be working on during these days. Depending on the size and complexity of the project, we might start a set of **cut-over activities** two days before going live, or a week, or sometimes more, in some rare cases. In addition to the data imports, clients will want to update their lists (Departments, Items, etc.), update inventory levels, validate trial balances, and more. The most important factor in this schedule is how long it will take to perform the client’s data imports.

Aside from the data migration steps, we would like to create a list of other steps we will need to take right before the go-live and make sure everyone knows their assignments as part of that planning. Each part of the system – ERP, CRM, the web store, the warehouse, payroll, HR functions, and so on – needs a checklist of last-minute steps to ensure that every list of values, every set of entities, and every set of forms and roles has been set up correctly in the production account, in time for going live. This is not the work of the implementation team for the most part; the client’s end users and managers need to own these tasks since they should be fully versed in how to use them and what they will be doing with them each day following the go-live date.

We typically use a list of standard records that have been imported into the account with NetSuite-led implementations for planning cut-over activities. But you can use whatever is easiest for your teams – spreadsheets, Post-it notes, whatever; just so long as everyone has their tasks assigned and project managers can monitor their completion. We need a concrete way of knowing whether we are ready to go live, just before that date. Has everything been verified by the SMEs who own each function? This is a *better safe than sorry* approach to going live. I have been a part of a couple of projects where the client just sort of assumed everything would work out, and then they had many unhappy surprises during their first week of using the system to run their business. To avoid that, we have to make sure everyone is on hand and completes their cut-over activities on time, in the last days before go-live.

Migrating inventory and trial balances

Related to the previous topic, for companies that manage physical items in inventory, we take special precautions to make sure NetSuite has the correct inventory levels for all of their items on the day they go live. To do this, we need to have practiced the data import in advance, for most or all of their items – getting the data from its current source, preparing it for import to NetSuite, and then performing as many imports as are needed. Sometimes, for instance, we get one data file from each warehouse location. On the day before the client goes live, we get an updated or “delta” file and run that import and then run reports in the system to confirm that we’re showing the same inventory count as the previous system did. This takes coordination between the NetSuite users and those familiar with the legacy system, but it’s a critical step in the cut-over for companies using inventory.

On the financial side of the business, we use the built-in **Trial Balance** report to compare balances on accounts right before go-live as well. We need to run a similar report in the legacy system, to see the current balance in that system after its transaction processing was frozen and compare the amounts to what NetSuite shows after our data is migrated in, but before we start recording new transactions. Most clients will consider getting the two systems’ Trial Balance reports to match a critical step in the cut-over. If the NetSuite report is different from the older system’s, we need to find the differences, identify the root cause of the problem, and decide whether or how we’ll fix the issues. Sometimes, it may be enough to just explain the differences, but in many cases, the data in NetSuite must be updated – quickly! – before the client agrees to go live.

Performing the final data imports before going live

When it comes to the final data migration push, we should only have a few things we need to import for most implementations – last-minute list value changes, item updates, and open transactions. Since they are usually ready ahead of time, all of the historical transactions should be in the production account before the final push, so this can be a fairly straightforward operation. We should have practiced importing the open purchase orders, sales orders, and so on into a sandbox before this, so there should not be any surprises. We are always prepared for something to go wrong, of course, but again, proper preparation should preclude any nasty issues with the data or the resulting transactions in the system.

The CSV import feature works on a *queue* basis, so as we begin these imports, it is important to remember to use whatever settings we found in our testing worked best for each record type. For instance, we may have found that since the account has 1 SuiteCloud Plus license, there are 10 available queues. And for most of our imports, we can distribute them across the queues, and also use the multi-threading feature, to get the highest throughput for the data coming into the account. But certain record types do not lend themselves to multi-threading, or we might have so many records of one type to be imported that we need to split up the files in advance and then use multiple imports running at the same time, or separately, to maximize the system’s CSV import resources efficiently.

It is important not to forget to review **CSV Import Preferences** in the account before you start performing your imports! You can find it by going to **Setup | Imports/Exports | CSV Import Preferences**. This is what that screen looks like:

Figure 20.2 – The CSV Import Preferences screen

These preferences include a setting that allows you to choose (globally) whether server-side scripts and workflows should be run against records being imported, which can have a huge impact on performance. Make sure that this has been set correctly, which will depend on whether or not you require scripts to update the new records as they are created.

The last step in this process, of course, is validating some of the data, to confirm that it still looks correct. This usually includes creating a few test transactions to be sure that, for instance, the customer data, addresses, items, segments, and everything else are all exactly as we expect them to be, and that all of the data works well together. We would not want to find out that we missed a custom category field's source list, for instance, which means that orders cannot be processed.

The final data migration process should be completed in production a few hours before we flip the switch on any implementation project. Hopefully, this is where all of your solid planning and practice pay off, and the go-live occurs on schedule, to everyone's relief.

Summary

Going live is a tough time for most clients. They have to change how the business runs every day. This is stressful in even the most well-planned cases. You can help the client through this by planning the data migration well ahead of the cut-over activities, as well as assisting them in making sure their historical transactions and other data will be brought into the account without any surprises. We do that by testing everything well in advance of the go-live date and including all of the relevant data types in that testing. We also make sure all the right people are involved in the data migration planning and testing so that nothing is left to chance. We also make sure we are there to help with any last-minute emergencies that might occur just before the go-live date.

At the end of the implementation project, once the data migration is completed and has been validated in the production account, and the business' users have completed all the last steps in the cut-over checklist, we go live. Users should have stopped entries from going into the legacy system a few days ago, and now they can start working in the production account. This is both an exciting and nerve-wracking time for these people, so you have to be there to support them however you can. Whether you do this on-site or remotely, regularly scheduled meetings and solid issue tracking will help smooth out the first few days. Assuming everyone did all the testing they should have in advance, this can be a very nice experience, where a few minor issues come up (no go-live has ever gone 100% smoothly), and you will deal with them quickly. A lot of last-minute training happens at this time too, of course, and new solutions are found to help solve any problems we did not anticipate. The client settles in and gets used to NetSuite in a way they could not before this. They then start to think about the next phase of improvements they want to make – with your help, of course.

I sincerely hope you learned what you hoped to from reading this book. Hopefully, you can go more confidently into your next client meeting, knowing a little (or a lot!) more about working with NetSuite clients than you did previously.

Creating this book was a labor of love for me, writing out all of the things I have learned about implementing NetSuite for you to read. NetSuite is a great system, and we always need more help working with clients and expanding the user base worldwide.

Self-assessment

As in the previous chapters, review these self-assessment questions and think about how you would handle each of these situations:

1. One week into a new client's implementation, you set up the first data migration call with their project leaders. You explain what your contract says is included in the scope of work for your team, which is just consulting on this topic. You start to explain what the client will need to do to ensure a successful migration, but the client gets angry and suggests that they were told you would be handling the data migration process. Their team will not have time for this, and they have no experience. How should you handle this?
2. Your team is helping a client with their migration testing in a sandbox account. They try to bring in a set of 100 customers, but a lot of errors are reported by the CSV import job. These errors are mostly about the phone numbers and email addresses being in an incorrect format, but many other problems have been reported. Without taking on the task of cleaning up the data yourself, what are some ways you can help the client with that task?

3. Two months before the go-live date, you and the client agree on a list of cut-over activities, and each of those has a name associated with it. Your team of consultants has a few assignments, but the client's managers and users are assigned the majority of these tasks. Three days before the go-live, they let you know that two people are off sick and that they need your team to step in and complete those people's assignments to stay on schedule. Should you help them? If so, what other steps do you need to take to make sure everyone understands the risks involved in you doing that?
4. Your software business client goes live on a Monday. At first, everything seems to have gone very well. Two days later, though, they notice that their GL reports are showing strange numbers. You realize that a customization is not performing as expected and now there are over 1,000 journals in the live production account with the wrong debit and credit amounts. You need to help them fix the bad data while the technical team updates the customization. What is the fastest option you have for managing these updates? How can you help the client ensure that they have caught all of the bad journals until that correction is in place?

Learn more on Discord

To join the Discord community for this book – where you can share feedback, ask the author questions, and learn about new releases – use the following QR code or link:

<https://packt.link/UytqI>



Appendix

My Answers to the Self-Assessments

In each chapter in this book, I have included questions to get you thinking about how you would handle various situations that might arise while you are helping a company through its NetSuite implementation. Here are my suggested answers to those questions, though these are definitely not the only valid answers. In most cases, you can solve problems in more than one way, and as long as you're thinking about the best solution over the long term, you should be fine.

Chapter 1

1. NetSuite's base tier allows most small businesses to run effectively on the system, but if a business processes more than the base number of transactions per month, they should talk to their NetSuite sales team about higher tier levels.
2. The easiest way to move something custom, such as a search or a role, is with the **Copy to Account** feature. Once it's enabled in both the source and destination accounts, any administrator can move the object with just a few clicks in the UI. The backup plan should just be to recreate the object manually in the destination account by carefully copying every setting from the source account.
3. The NetSuite **Sales** team handles new customer requests and will guide every new client to just the right set of features and optional modules to serve their business.
4. **SuiteCommerce Advanced (SCA)** is an e-commerce web store that allows customers to purchase items, manage their accounts, and more. **SuiteCommerce InStore (SCIS)** is a retail store point-of-sale system, based largely on SCA, but with additional features that make using it in a store, at a register, easier.
5. For tracking time and expenses, I'd suggest that the business check out the native **Projects** feature and tie timesheets and expenses to them. If there's a need to make it easy for workers to remotely enter their time and expenses, they should also investigate the NetSuite mobile app, for iOS or Android.

Chapter 2

1. Have you ever tried this and then had to adjust the approach mid-stream? (That's not fun; I don't recommend anyone try this.) There will always be new ideas and approaches to try, and tweaking or fine-tuning your methods should be part of any business's continuous improvement program.
2. This is where your soft skills, such as good communication, play a very important role. For instance, when dealing with changes, you have to find a way to both remind the client of the scope in your current contract and also suggest a way forward to handle their change requests – and you need to do that without upsetting them!
3. You might have guessed that I would say upfront that I am very strongly in favor of transparency with clients.
4. There's no single right answer for every client or implementation team. Knowing what your contract says and what your company's policies are in advance is key, though, so you know how much flexibility you can have in working out solutions that will work for everyone.

Chapter 3

1. On just one occasion, I worked on a project that went exactly according to plan, all the way through. I realized this was happening just before we wrapped it up and found myself amazed at my team and the client too! What worked is just what you would expect – we all stressed good communication throughout the project and the client was well organized and stayed focused throughout the project.
2. In my mind, it would be wrong to tell a client that they're not allowed to change the path of the implementation. We do want to keep the originally stated goals for the project fresh in the client's minds, but as long as they do that, we can adapt the project as they need.
3. We don't always do this in my projects, but I think it's the right idea, as long as things haven't changed in the project so much that the original goals no longer make sense.
4. We generally avoid this at NetSuite. It generally leads to unrealistic client expectations and problems for the team trying to complete an implementation.
5. The simplest answer is that you need to make sure those change requests aren't ever being communicated between the client and a consultant. Both project managers need to know about all the changes as they appear so that they can add them to the list and prioritize them, and so on. Then, we let the chips fall where they will.

Chapter 4

1. I think this company could be described as being in the retail industry, primarily.
2. Once, I worked with someone who was meant to be the NetSuite evangelist within their company, but they resented that their company had chosen the product over another competing platform. They made every step in the implementation more difficult than it should have been. I worked with them to help them see how NetSuite could help them every day. Near the end of the project, we had to have a meeting with senior managers to confirm that the company was going live on NetSuite and that there were no serious problems with doing so. From that point forward, this person helped us instead of fighting us at every step.
3. We avoid questions such as option 1 since they tend to lead us in directions we don't want to go. The question in option 2 is much more targeted, and so better in the long run.
4. How you document your requirements is up to you – just make sure they're always easily understood by your clients, and in fact, confirm that with them before moving on to configuring the account or creating any custom things to meet requirements. It will help if you have a standardized template with the relevant sections included for each requirement you document.
5. Generally, I prefer to try to find a pre-built SuiteApp solution for any new client requirements, as opposed to creating something custom from scratch. As long as the client's requirement is something common in their industry and not too complicated, there is often something out there we can use. Of course, that's not always true, and so we must be prepared to create something specifically for the client when the need arises.

Chapter 5

1. We usually recommend they apply the roles that are available in their new NetSuite account to their user groups, without a lot of modifications. This is usually not a great challenge and keeps things moving at the beginning of an implementation.
2. By the time a user can make a request like this, you and the client should have already defined which people will get the administrator role and why they were chosen. You should be able to tell John whether he meets the criteria for having that role, and if not, help him understand how he will still be able to complete his work, although occasionally he may have to ask someone else with the administrator role to help him with that. Refer him to the actual administrators within the company for follow-ups.
3. For the football pool request, I would suggest they only add a small set of fields directly to the employee record. They won't be able to track multiple years with this, but they really shouldn't need a ton of history in NetSuite for this anyway. It's our job to always look for a simpler solution than the one a client initially asks for, whenever we can think of one.

4. We usually aim to have all of the roles in an account set up properly and assigned to all of the people working there by the day we officially start user acceptance testing (UAT). We want people to test their daily functions with these roles in place, limiting them in some cases and empowering them in all others. One disadvantage of setting up roles later on is that then, some people may have more permissions available early on than they will at go live, and that can cause confusion when the permissions change.

Chapter 6

1. Every NetSuite client starts with just one account, and 99% of them should stick with just that. There are many controls available within an account that can limit users, transactions, and so on to just one related subsidiary. I've only heard of businesses needing multiple NetSuite accounts when their transaction volume was such that they needed to split up the processing of those records across multiple accounts. This is a very expensive option that most businesses don't need and won't want to pay for.
2. The folks who are acting as the client's account managers, that is, their sales reps, need to know as soon as possible when a change or new requirement from the client might have to be purchased to make it available to them. They can have one of those wonderful commercial conversations about what the new features will cost, how they're supported, and so on with the client next.
3. Adding a new department is easy enough, but you might also need to make it easy for them to create assembly items and so track the assembly work via work orders and assembly builds as well. Keep these transactions as simple as possible, just allowing the client to track the costs involved in the process to the extent they need to.
4. When disagreements arise on a call, especially when they're solely between the client's users and managers, it's a good idea to let them talk it through for a minute or two, hoping a quick resolution can be found, but then find a courteous way to suggest that maybe the subject should be moved to the parking lot and discussed later, on the client's side only, and then a decision can be reached separately.
5. NetSuite's **Help** and **SuiteAnswers** features are goldmines of useful information, all easily searchable and located in a variety of ways. If you know what topic you want to explore further, as was the case in this scenario, go ahead and pop open either screen and refer the folks you're working with to these options so that they can get more information following your meeting.
6. Budgets are powerful tools for growing businesses, as long as they're used properly – and frequently. If an executive thinks they can manage their business's growth without a budget, we can't force them to try to use the feature, but we should always at least explain the advantages and make sure all of the top-level stakeholders we're working with agree on that one person's approach before shelving the idea. It's perfectly fine if they're using a tool outside of NetSuite for this, of course.

Chapter 7

1. NetSuite allows us to first create a customer record and then use an easy-to-miss link on the **Relationships** tab, creating a vendor for the same entity.
2. Project templates allow us to define certain parameters around a project type in advance. This includes the types of resources that will be staffed and the tasks that usually make up the project. These details can be modified for each new project we create from a template.
3. You could create a sub-customer record for each of these people via a fairly quick CSV import. This would allow the client to enter the name of any sub-customer on an order but then record payments as coming from their parent company. NetSuite puts a hard limit on the number of sub-customers any one parent can have, though – 10,000.
4. Yes, a contact can be associated with more than one customer record.
5. The furniture company should use the **Matrix Inventory** type items for this scenario. They will be able to define all the color and fabric options for each matrix parent item, and then shoppers in the web store will be able to easily make selections from those varieties.
6. Two million items is a lot, but NetSuite can handle it – if the client really needs it to. But first, work with the client to validate the items list to make sure they're not misunderstanding how you want them to define items as unique purchases or sales. Also make sure their list has no duplicates. Counsel them against importing items “just in case” they might need them, whenever you can.

Chapter 8

1. Talk over how they process these “commitments” in their current system, and you will find the perfect match in NetSuite (opportunity, estimate, sales order, etc.).
2. Generally, NetSuite doesn't differentiate journals by who entered them. One idea for handling this might be to create a new custom transaction of the journal type and only give the senior accountant permission to create or view these records. Of course, you want to make sure that the client's other senior people know this is going to happen and approve this course of action.
3. With high transaction volumes, we want to know a few things. What are the average number and the maximum number of lines on these orders? How will these orders be created in the account? (Such a high volume suggests something has been done via some form of automation – CSV import or integration, for instance.) Which of the customizations you're planning in the scope of the project need to run for these orders as they're created?
4. There is no native NetSuite feature that can combine the line items from multiple quotes into one new sales order. Depending on the client's requirements around exactly how those lines should be combined, a script might be developed to meet this need.

5. If the services company creates the projects first, they can then format their sales orders to include a link field and thereby link every line on every sales order to a project. If they want this to be automated, a workflow should be created to set a default value, based on various inputs.

Chapter 9

1. Generally, the idea of defining thousands of unique kit items with overlapping sets of components seems risky. Without proper maintenance and controls over that process, we could see the volume of data NetSuite is storing explode into an unmanageable mess. One alternative idea would be to set up a couple of custom record types, with one record type defining a custom kit item, and the other record type holding the list of component items used in that kit. Each of the kit items might have an expiration date, allowing the client to say that it shouldn't be used after a certain time has passed. With this, you could allow the client to define the lists of components more fluidly than native NetSuite does, and you could create simple automation to purge expired records from the account once per month (for instance).
2. We can set up two very different forms for use with these contract and non-contract sales orders and train users on when to select each form.
3. One approach I've taken in the past is to define a custom free-form text field for storing the PIN but then use a script to essentially encrypt that value and replace the plain text PIN with the encrypted version in the field. With this, the user will still be able to edit the PIN field whenever they need, but as soon as they save the transaction, nobody will ever see the plain text value in that field again.

Chapter 10

1. The Sales Center in NetSuite is really just a different *skin* on the native NetSuite UI. You can customize what sales reps will see via the standard SuiteBuilder elements, including forms and records, and fields as you need.
2. NetSuite's **Calendar**, **Events**, and **Activities** features are not very commonly used, but using them for this scenario makes a lot of sense. The calendar data will be stored in NetSuite, the central hub for the whole company, and custom views can be created for the calendar, making it easy to use by people who are not otherwise very NetSuite-savvy.
3. This would be a great case for creating an entire center, list of categories, and links for this warranties-and-returns solution to occupy. You'll be able to add shortcuts and other links to its contents to people's dashboards, too, to make it even easier to work with those custom features.
4. NetSuite allows us to control which transactions are visible in the customer center for our client's customers.
5. One idea would be to create a search that includes a dynamic filter for the current user and lists just the sales orders waiting for that user to approve them. Once the search is defined, you could add that to the dashboard published to the sales manager's role.

Chapter 11

1. NetSuite offers a special type of item for downloadables, which makes it easy to offer them from your SuiteCommerce or other web stores.
2. Yes, the system allows us to control which fields are shown in that popup from the form's definition. You can use the **QuickView** tab on a custom entry form to control this.
3. One way to help the warehouse manager with this would be to create a custom version of the forms for each item type, with just the fields they want to be able to edit.
4. We would usually create these items as non-inventory items, and we'd enable the **Drop-Ship** checkbox at the same time.
5. Yes, when we have just a few options to make available on sales orders (for instance) for a given item, we can use **Item Options** to define that list (on the item record), and then users can select from that list as they add the item to opportunities, quotes, sales, etc.

Chapter 12

1. You could create a price level for this and just set the prices on those items at that level when it's appropriate, but this wouldn't restrict the prices to certain customers. To do that, you need to use the **Item Pricing** tab on each of those customer records, assigning just the right price for those 1,000 items as you need. You can use the CSV import feature to manage this easily as well.
2. NetSuite does support multiple levels of parent and sub-customer records. We can define this nesting up to nine levels deep, in fact, but generally, we'd reserve that for the rarest cases where it really makes sense to do so. Managing that many levels of customers would be relatively difficult, in my experience.
3. You could quickly show them how to open and update cases in NetSuite, but it might be simpler if you created a much less field-heavy custom record type, associated with the customer record for a starting point. Give that record list just a customer field, a status, and a big **Notes** field to get the client started and let them know that they can migrate that data into the native case record when they're ready to after going live.
4. Native event records already have fields for linking them to companies (customers or vendors), contacts, and transactions.
5. To update a lot of customer records quickly, you could either set up a CSV import for this, with just the **Customer ID** field populated in the file and the **Sales Rep** field set to a default of the new value, or you could use a mass update to make the same change based on a search's criteria.
6. Check out the new **View and Approve** role types for users who need access to just a couple of records and only need to be able to view them and then approve or reject them via a workflow.

Chapter 13

1. We rarely set up separate NetSuite accounts for one business. The users will be able to set up as many subsidiaries as they have in their single account and report on them as they need to.
2. Since the native exchange rate feature tries to update the rates every day, you'll have to find a custom way to update the rates in the account on an annual basis. This is generally not a complicated customization; for instance, if your client has a preferred service, the rates can be gathered via the internet.
3. Scripts written using the NetSuite-provided GL plugin can make updates to the GL impact on transactions. They do not ever change the GL impacts NetSuite wants to apply, but they can then apply additional updates following the native account updates.
4. Information on setting up custom payment files can be found on the help page entitled *Creating Custom Payment File Templates*. You can find help on setting up bank statements in a non-standard format via the page entitled *Creating a Bank Statement Parser Plug-in Script File*.
5. Two settings can help with this situation. First, NetSuite allows us to assign permission to a role called **Override Period Restrictions**. With this role, a user can edit transactions in closed accounting periods. There is also a setting on each period called **Allow Non-G/L Changes**. If this is enabled, users with the aforementioned permission will be able to make changes to transactions as long as the change doesn't change the transaction's GL impact. With either of these options enabled, you should make sure the client understands the impact of changing transactions in closed periods in financial reports and other reporting and that they know what their company's policies are for this sort of thing.

Chapter 14

1. A normal PO will suffice for this situation, although you'll use it to purchase service items instead of inventory or other item types.
2. This seems like a good case for using a special order in NetSuite since the company won't be purchasing the customized item for most of its customers.
3. A vendor prepayment in NetSuite can be set up for use with either a single purchase order or none, but it can't be linked to a set of POs. This is because the system will try to auto-apply the prepayment to the PO when the vendor bill is recorded later on.
4. The user can edit the vendor bill in question and select the **Recalc** button there to correct the *Transaction does not balance* error.
5. One option would be to just train the users on the native screens and customize the forms to make sure they're as simple as possible. Another option would be to create a custom script that auto-creates a return from selected lines on a PO, but that should not be your go-to solution in cases such as this.

6. To automate the import of bills from vendors, suggest that they check out the new **Bill Capture** tool from NetSuite or any of the equivalent offerings available in the SuiteApp Marketplace. There are quite a few apps listed there now, all vying to streamline A/P processes for our clients. To help with paying bills, suggest the client consider using **Electronic Bank Payments**.

Chapter 15

1. The client whose orders consist of various physical and service-type items could use quotes to record the sale before it's finalized with the customer. With this, they can track how confident they are in the sale and other similar details, up until they're ready to make the sale official. They could use the Order Guides SuiteApp from NetSuite to have the system remember sets of items that are frequently sold together.
2. With unusual requests such as this from a client, the first thing you must do is to understand all of the ways these items and their sales differ from the sales of physical items. In this scenario, I might suggest that the school use non-inventory items for setting up digital books. They can sell them via sales orders and so on, or they could use the **Download** item type, if they want to be able to sell and allow users to download the books themselves, directly.
3. First, set up the shipping items on that screen, enabling whatever cost amounts are appropriate. Then, enable the **Charge for Shipping** preference and set a shipping cost on each item as appropriate. On sales transactions, users can then use promotions or manual edits to control the shipping charged to the customer for each order.
4. I would suggest checking out the native **Invoice Grouping** feature. It's easy to enable and users can be trained on its configuration and use in a relatively short time. Of course, a client should then test it quite a bit as well, with a wide variety of invoices, to ensure they're getting the right results prior to going live, if possible, or immediately after, otherwise.
5. When this kind of request comes from a client, first, we like to understand exactly which of the steps that we've shown the users seem like they're onerous. Once we clearly understand where the complaint is coming from, we can usually make tweaks to the records or the process, to make them more user-friendly. For instance, if the business managers and accountants are OK with this, we could have the users skip creating return authorizations in some cases and go straight to creating credits only. If the users want even more of the process to be streamlined, we might offer to create a script that can automate some of the steps in the process.

Chapter 16

1. First, as always, we start by understanding the process the users follow and the client's requirements for tracking that gift basket assembly work in NetSuite. If it's clear they need to do more than just record the outcome of the work, then we can enable the manufacturing transactions (**Work Order** and **Assembly Build**) and train the users on their creation, maintenance, and reporting. If we can get away with something simpler, though, a custom transaction might just suffice.

2. They could use **purchase contracts** to track the expected per-month flat rate they'll pay the truck rental company in advance. Or they can just record the expense on a purchase order as it occurs. They might like to create custom transactions, though, if they want to see these rentals show up separately on their financial reports.
3. Generally, since a feature such as **Knowledge Base** is very simple and most clients can learn to use it on their own, we might just point the system's admins to screens, give them a quick tour of the features available there, and then let them run with the setup and configuration and import Knowledge Base articles on their own.
4. A custom transaction type would be a good fit for this scenario since having this would keep the transactions separate from native transactions and the client could control access and perform reporting separately, as they liked. An alternative idea, though, would be to just add a custom field to the native transactions (purchase orders, for instance) and have that field set to a specific value when the record represents one of these "beta" transactions.

Chapter 17

1. You could set up a saved search, using the client's criteria, and then schedule it to run as often as needed. On the **Email** tab of that search, you can set up the email settings so the results will be sent out each time it is executed.
2. *SuiteAnswer #28923* describes a way to create a saved search to replace this report, with the filters set up so that you can split up the details just as required for this scenario.
3. To share a workbook on a set of users' dashboards, first, create the workbook and set its permissions as you want so that the roles have access to view it. Then, set up the dashboard so it includes a portlet with the workbook enabled and configured as you like.
4. To send the client's data immediately to their external partner, consider creating a server-side script that runs as those records are created or saved searches are executed, and sends the data out. For records that can be sent in batch mode, use the **SuiteAnalytics Connect** service, and write queries to pull sets of records back from NetSuite into the external service.
5. This can be a challenge in a test environment because we don't have all of the client's users working to create new data yet (as we will once they go live). We still need to test the performance of connections like this, though, so explore how you can create a lot of test data in a short time via tools such as **CSV Import**. Use Excel to help auto-populate records with repeating values, for instance, to save time in creating lots of similar but unique-enough records for this kind of test.
6. SuiteQL relies on roles and permissions to access the data we report on. The developer should be able to work with your client's administrators to amend the role they're using to give it all of the permissions it needs.

Chapter 18

1. This is a good case where just knowing how all of the native features work and what options they support can help you avoid talking about customization with a client. In this case, NetSuite supports an auto-apply feature that works just as described, and users working on the payment screen can use that whenever they like by clicking a button.
2. Yes, you can create a **SuiteFlow** that will use the value in the custom gross profit field to set the value of the native commission field, based on whatever calculation you need to use.
3. For this charge record customization, **User Event** scripts would be a good choice because they run as soon as the charge is saved into the system, and NetSuite will perform the updates you need quickly at that moment. There's no limit to the number of records that could be saved and processed by a **User Event** script in this context. The only disadvantage to this type of script is that users will always have to wait for it to complete its process every time a charge is created. On the other hand, a **Map/Reduce** script (which works in batch processing mode) could be scheduled to run late at night. Users won't see the updates made by the script while they work; but, at the same time, they won't have to wait any extra time for that processing to occur either.
4. The simplest way to include the customization documentation in NetSuite would be to just save the files into the file cabinet and tell users where to find them. You could go a step further, however, by creating Knowledge Base articles, one for each custom solution you implemented, documenting the details directly in that record, and also attaching the relevant document to the same place.
5. Deploying custom forms is always tricky in that they contain a lot of settings, and users expect them to work exactly the way they've tested them prior to going live. One way to avoid upsetting the client would be to use **SuiteBundler** or the **SuiteCloud Development Framework (SDF)** features to deploy the forms to the production account with unique names and IDs. This way, the client's team can compare what you imported to the forms they already have in the account and update their forms to account for any differences they choose to keep.

Chapter 19

1. First, research the external expense tracking system and ask the company that makes it whether they already have any integration with NetSuite. If they do, it should be available as a SuiteApp, so use that. If they don't, then look for other manual processes the client's users could follow to share data between NetSuite and the expense app. In many cases, if the volume of data is not too high, users can quickly find a way to use CSV files to move data back and forth as needed – and so still go live on schedule.

2. When a client indicates that an integration will need to support a very high volume of data in its transfers, we need to ask questions about whether that data is really needed in NetSuite (you'd be surprised at how often it turns out not to be needed), and what purpose it will serve in the system. Assuming it is needed, then we ask questions about how often these transfers will take place and how many NetSuite records will be created in each case. If these are transactions, we ask about both the number of records and the number of lines on each. We can then look for the best-performing way to meet the integration requirements.
3. You'll still need to work with the five users, get them used to performing their tasks, and so on, but otherwise, this implementation will be all about the integration. Most of your conversations will be technical in this scenario, but you'll still need a combination of functional and technical expertise on your side to make sure that as all of the integration's development and testing take place, the accounting team will be happy with the results.
4. With a little practice, we can use Excel and the NetSuite CSV import feature to create a lot of test data in an account in a short time. In this scenario, you just need to create enough data to fulfill the "every 15 minutes" requirement, so work with the client to figure out how many transactions that might be and create those when you're ready to run the SuiteAnalytics Connect test.
5. When a client asks for help that your company doesn't have expertise in, it's usually enough for you to point them in the right direction. In this case, that might mean helping them understand the skill set they're looking for so that they can then begin a search online or among their peers for another developer who can help them finish their project.

Chapter 20

1. Generally speaking, this is why we have contracts in the first place – so that misunderstandings should either not ever happen or be very rare. If the sales team member who drafted the contract is still available, set up a call with them and the client and see whether the wording in the contract can be straightened out. If that's not an option, work with your company's managers to come up with an alternative plan that will get the client what they need – and charge them whatever is appropriate for the additional work.
2. It's a good idea to give the client a data migration planning document before the actual data import work begins, and that usually includes formatting tips for fields such as these. Beyond that kind of proactive assistance, you can also suggest that the client report each error to you, and then you can provide a specific idea of what they need to change in their source data to fix those errors. Encourage them to work on a process they can apply to all of their import files from that point forward, to avoid seeing these errors from the CSV import process. Of course, you could be more hands-on if you choose and review their import data yourself before it is imported.

-
3. Generally, as long as you don't think the client is intentionally avoiding doing the work themselves, you should work with your company's managers to find a way to assist the client and get them live. There is some risk in your team helping with this, especially if they'll need to make configurations or other choices in setting up a production account that they're not familiar with. Make sure the client is aware of each and every decision that is made this way so that their users can follow up and make corrections/changes where needed, soon after going live.
 4. If your team was responsible for this customization, you need to help the client clean up the database if that's required. You can look at using the **Mass Update** feature, with a saved search targeting the incorrect records. But in most cases, with a transaction such as journals, it will probably be faster to create a search first, export those results, change the data in Excel (or any other spreadsheet application), and then use a CSV import to update the existing records in the account. Use that same search (with its carefully designed criteria) to look for any new records showing the same problem for as long as you have to until the customization is updated to resolve the issue once and for all.

Index

A

- accounting** 66
 - basics, setting up 76, 77
- accounting periods**
 - managing 178
- Account Reconciliation** 83
- accounts payable (A/P)** 66, 83, 104, 183
- accounts receivable (A/R)** 66, 77, 195
- Acme Industries** 54, 55
- actions** 247
- administrators** 66, 69
- Advanced Customer Support** 10
- Advanced Intercompany Journals**
 - (AIJs) 173, 175
- Advanced Inventory Management** 13
- Advanced Manufacturing features** 107
- Advanced Receiving** 83
- Advanced Revenue Management**
 - (ARM) 13, 56, 176, 217, 218
- Advanced Revenue Management**
 - (Essentials) 218
- Advanced Revenue Management**
 - (Revenue Allocation) 218
- agile approach** 23, 26-29
- amortization templates** 219, 220
- assemblies** 98
- assembly items** 148

- automation**
 - gaps, identifying 244
 - needs, identifying 244
- Avalara AvaTax** 14

B

- banking features**
 - setting up 172
 - using 172
- basic projects** 92
- Bill Credits** 193
- blanket orders** 198
- Blanket Purchase Orders** 83
- Boomi** 14, 276
- bulk orders** 198
- bundles** 98
- business processes**
 - Call to Resolution (CTR) 102, 107
 - Design to Build (DTB) 102, 107
 - Hire to Pay (HTP) 102, 107
 - Lead to Quote (LTQ) 102, 107
 - Marketing to Return on Investment (MTR) 102, 107
 - Order to Cash (OTC) 102
 - Procure to Pay (PTP) 103
 - Project to Delivery (PTD) 103, 108

Record to Report (RTR) 102
Return to Credit (RTC) 103, 108
revenue recognition 108
Web to Order (WTO) 103, 108

business requirements documents
 (BRDs) 60

business-to-business (B2B) 48

business-to-consumer (B2C) 48

C

Call to Resolution (CTR) 102, 107

carve-ins 219

carve-outs 219

case record 207

cases 106, 220

cash sales 197, 200

Celigo 276

centers 114, 129

change management 77

change management, for custom
 objects 124, 125

 issues 125

 requirements 125

 system change 125, 126

 use cases 125

change order (CO) 58

Chart of Accounts (COA) 7, 78, 79, 272

check 189

classes 82

classification 80

client scripts 251

competitors 96

 setting up 161

conference room pilot 38

contacts 93, 94, 158

Continuous Integration/Continuous
 Delivery (CI/CD) 256

Copy To Account 6, 256

credit memo 207

CRM role 163

CSV Import feature 78, 98, 168

currency

 exchange rates 171, 172

currency lists 170

currency management

 implementing 169

Customer Center 12, 93, 131

 setting up 135, 136

 using 134

Customer Deposits 204, 205

Customer Payments 206

customer rebates 106

customer record import

 bad addresses/phone numbers 156

 duplicate records 156

 invalid records 156, 157

customer refund 207

Customer Relationship Management
 (CRM) 3, 11

customers 90-92

 setting up, in account 154, 155

customer service representatives
 (CSRs) 125, 220

custom exports

 creating, from NetSuite 267, 268

custom forms

 actions 120

 fields, adding 118

 fields, rearranging 118

 fields, removing 117

 for entry 115, 116

 for transactions 115, 116

- related features and settings,
 - enabling 118, 119
- roles 120
- storing, with record 120
- customization requirements and solutions**
 - documenting 253-255
- customizations**
 - managing 255, 256
- custom lists and fields**
 - checkbox, versus list 123
 - defining 122
 - image, versus inline HTML 124
 - single-select lists, versus
 - multiple-select lists 123
- custom record types**
 - used, for storing custom
 - record types 120, 121
- custom segments** 82, 83
- custom transactions** 222, 223
- cut-over activities**
 - planning 277

D

- dashboards** 129
 - setting up, for groups by role 136-138
- data imports**
 - final data imports, performing
 - before live 278, 279
- data integration, managing** 264, 265
 - Middleware 266
 - SuiteScript RESTlet 267
 - SuiteScripts 265
 - SuiteTalk REST 266
 - SuiteTalk SOAP 266
- data migration process**
 - planning 272, 273

- data migrations**
 - data, testing in phases 276
 - imports, performing for testing 274, 275
 - performance 276
 - reference link 274
- datasets** 232
 - linking 235
 - utilizing, for deeper analysis 232-235
- Deleted Records** 78
- departments** 81, 82
- deployment** 125
- deployment document** 255
- Design to Build (DTB)** 102, 107
- distributors** 50
- Documents Center** 134
- donors** 91
- Drop Ship Orders** 187, 199

E

- E-commerce Sales Orders** 199
- e-fix** 8
- Electronic Bank Payments** 192
- Electronic Funds Transfer (EFT)** 192
- elimination subsidiary** 176
- eMerchant services** 14
- Employee Center dashboard** 12, 131, 132
- employees**
 - assign roles 72
 - creating 71, 72
 - expense limits and approvals setup 73
 - grant access 72
 - managing 71, 72
 - supervisor setup 72
- enabling features** 76, 77
- Enterprise Resource Planning (ERP)** 3, 11, 34, 91
- entities** 90

Estimates 195, 197
expected receipt date 186
Expense Allocation Schedules 173
Expensify 14
exports
 creating, with SuiteAnalytics
 Connect 236-238
External ID 273
Extract, Transform, Load (ETL) 261

F

fair market value 219
Field Service Management 93
Financial Accounting Standards
 Board (FASB) 200, 217
 URL 200
Financials First 12
five Ws 41, 106
Fixed Assets Management
 (FAM) 85, 173, 174
 reference link 85
fixed-bid contracts 24
For Purchase transaction 184
For Resale transaction 184
functional designs 253
functional requirements
 documents (FRDs) 60

G

general ledger (GL) 35, 60, 78, 79, 200
 setting up 168
Generally Accepted Accounting
 Principles (GAAP) 200
generally available (GA) 266

global search 226
 tips 226, 227
groups 98, 161

H

happy path 38
Hire to Pay (HTP) 102, 107
hot pushes 8

I

identity provider (IdP) 71
Inbound Shipment 83, 149
 utilizing 187
information technology 67
integration long-term planning guide 263
integrations 260
 data preparation for 261
 documenting 262-264
 flow 260
 information gathering 262-264
IntelliChime 54, 56
intercompany accounting 169
Intercompany Journal Entries (ICJs) 175
International Accounting Standards
 Board (IASB) 217
inventory
 migrating 278
inventory-level warnings 70
inventory management
 options 141-143
Inventory Transfers 203
Invoice Groups 204
invoices 203, 204
issues 106, 220-222
issue tracking system 58

item form

with Fields list 143, 144

Item Fulfillments 149

managing 201-203

item groups 148**item options**

setting up 146, 147

Item Receipt, 149

processing 188

items 67, 90, 96-98

purchase prices 145

sales pricing 145

special-purpose items 98

item types 147

assembly items 148

description 147

discount 147, 148

enabling 141-143

item groups 148

kits 148

other charges 149

subtotal 149

J**JavaScript 248****joins 232****Journal Entries**

advanced intercompany 175

elimination 176

normal intercompany 175

revenue recognition 175, 176

reversals 174

statistical journals 176

using 173

K**Kanban 26****Key Performance Indicators (KPIs) 150****kits 98, 148****knowledge base 67, 106****L****large language models (LLMs) 122****lead nurturing marketing campaigns 91****leads 90-92****Lead to Quote (LTQ) 102, 107****Learning Cloud Support subscription 10****less-than-truckload (LTL) 55****limited-access centers 12****locations 80, 81****lot numbered items 98****M****manufacturer 48****Manufacturing (MFG) 11, 48****Manufacturing Mobile 18****Manufacturing Scheduler 216****map/reduce scripts 252, 265****Marketing to Return on Investment
(MTR) 102, 107****Matrix Item Assistant 146****matrix items 99****matrix item types**

setting up 146, 147

methodologies 22, 23

adapting 30, 31

agile approach 23, 26-29

applying, to project plan 39, 40

purposes 22

waterfall method 23-26

Microsoft Excel Web Query 232
micro-verticals 50
migration 125
Multi-Book Accounting 83
Multiple Currencies feature 169
multi-tenant approach 4
MyAccount 11

N

native centers

Customer Center 131
Employee Center 131, 132
Partner Center 132
setting up 130
Vendor Center 132, 133

NetLedger 4

NetSuite 4

editions 8, 9
mobile apps 18
people 10
product offerings 11-15

NetSuite Account Reconciliation (NSAR) 178

NetSuite accounts 5

data 6, 7
Development accounts 6
Production accounts 5
Release Preview accounts 6
Sandbox accounts 6
Sandbox refreshes 7, 8

NetSuite Analytics Warehouse (NSAW) 13, 239

NetSuite Bill Capture 190

NetSuite Connector 49

NetSuite Customer Center 158

NetSuite Mobile 18

NetSuite Planning and Budgeting (NSPB) 13, 85

NetSuite processes

customizing, with workflows 245-248

NetSuite Service Tier Structure

reference link 5

NetSuite Statistical Accounts 169

NetSuite SuiteCloud Platform 15

SuiteAnalytics Connect 17
SuiteBuilder 16
SuiteFlow 16
SuiteScript 16
SuiteTalk 17

NetSuite SuiteCommerce InStore 13

NetSuite Support 71

NetSuite WMS 11, 12

nexus 180

non-governmental organizations (NGOs) 48

normal intercompany journals 175

O

OAuth 262

One Implementation Bundle 61

OneWorld product 11

on-premises applications 4

OpenAir 13, 50, 163

Opportunities 195

Oracle CX Cloud 13

Oracle Integration Cloud (OIC) 266, 276

Order Guides 198

orders 198

Order to Cash (OTC) 102, 145, 195

requirements gathering 105, 106

organization

requirements documenting 60, 61

organizational structure

reference link 66

organization, business and people 51, 52

- project helpers 54
- project manager (PM) 52
- subject matter experts (SMEs) 53
- users/testers 53

organization, industries 48

- manufacturing 48
- miscellaneous 50, 51
- not for profit 48
- retail 49
- services 49
- software 49
- wholesale/distributor 50

other charges item type 99, 149**P****parent customer** 94**Partner Center dashboard** 132, 161**partners** 96

- setting up 161

Payment Card Industry (PCI) 206**period close** 177

- managing 178

period restrictions

- overriding 178, 179

Point-of-Sale (PoS) system 13, 49, 197**portlets** 137**portlet scripts** 137, 253**Procure to Pay (PTP)** 94, 103, 145, 159, 183

- requirements gathering 104, 105

project 162

- managing 162

project deliverables 199**project goals**

- examples 35
- setting 34, 35

project helpers 54**Project Management** 92, 162, 163**Project Managers (PMs)** 28, 35, 52**project plan** 33

- change requests 41, 42
- elements 34
- methodology, applying 39, 40
- tasks, defining 36, 37

Projects features 92, 108**Project to Delivery (PTD)** 103, 108**prospects** 90-92**pseudo-code** 253**Purchase Contract** 186**Purchase Orders (POs)** 184-186, 199**Purchase Requests** 184, 185**Q****queries** 237**QuickBooks** 4**R****Rebates and Trade Promotions features** 106**records**

- grouping, by business process 101, 102

Record to Report (RTR) 102

- requirements gathering 103

Redwood 115**Release Preview (RP)** 5**reports**

- creating, for in-depth analysis 229-231

request for proposal (RFP) 52**request for quotation (RFQ)** 52**requirements analysis process** 57

- business process reviews 58
- conference room pilots 58, 59
- demos 58, 59
- go-live 59

- project kickoff 58
- UAT 59
- walk-throughs 58, 59
- requirements and design document 253**
- requirements traceability matrix 60**
- Requisitions 184, 185**
- resources 93, 162**
 - managing 162
- responsibility assignment matrix 52**
- responsible, accountable, consulted, and informed (RACI) 52**
- RESTlet 252, 267**
- retail 49**
- Return Authorization 203**
- Return on Investment (ROI) 14**
- Return to Credit (RTC) 103, 108**
- revenue allocation 219**
- revenue arrangements 219**
- revenue recognition 108, 173, 175, 176, 200, 216, 217**
 - plans 219
 - rules 218
- Reversal Journal 174**
- RF-SMART 14**
- roles 67-69**
 - custom records access 69
 - dashboard 70
 - forms and searches 70, 71
 - reports access 69

S

- sales 67**
- sales orders 197**
- SAML 2.0 71**
- saved searches 227**
 - features 227, 228
 - limitations 227

- scheduled scripts 252**
- script 248**
 - client scripts 251
 - creating 249-251
 - map/reduce scripts 252
 - portlet scripts 253
 - RESTlet 252
 - scheduled scripts 252
 - Suitelet 252
 - User Event scripts 251
- Scrum 26**
- Scrum Master 27**
- SDF Project 256**
- searches 226**
 - global search 226
 - saved searches 227
- secure file transfer (SFTP) 265**
- segmentation 80**
- serial numbered items 98**
- service deliverables 199**
- Ship Central 202**
- single sign-on (SSO) 71**
- software-as-a-service (SaaS) 219**
- Special Orders 186, 199**
- special-purpose items 98**
 - lot numbered items 98
 - matrix items 99
 - other charges 99
 - serial numbered items 98
- sprints 26, 27**
- SQL-92 237**
- Standalone Invoices 203**
- Statement of Work (SOW) 24, 58, 264**
- states 246**
- statistical journals 176**
- strategy document 273**
- StrongPoint 15, 255**
- sub-customers 93, 94**

subject matter experts (SMEs) 38, 53, 89
Sublist Action Group 247
subsidiary 80
SuiteAnalytics Connect 14, 17, 236, 267
 exports, creating with 236-238
 setup screen 237
SuiteApps 14, 15, 69
 Avalara AvaTax 14
 Boomi 14
 eMerchant services 14
 Expensify 14
 RF-SMART 14
 StrongPoint 15
SuiteBilling 12, 49, 56
SuiteBuilder 16, 114, 245
 features 114, 115
SuiteBundler 256
SuiteCloud Development
 Framework (SDF) 8, 256
SuiteCommerce 11, 138
SuiteCommerce Advanced (SCA) 11
SuiteCommerce InStore 49
SuiteCommerce web store 158
SuiteFlow 14
Suitelet 252
SuitePeople 12, 72, 136
SuitePromotions 147, 200
SuiteQL 238
SuiteScript 14, 16, 248
SuiteScript RESTlet 267
SuiteScripts 265
SuiteTalk 14, 245
SuiteTalk REST 266
SuiteTalk SOAP 266
SuiteTax feature 179
Supply Chain Control Tower
 (SCCT) 149-151

Supply Change Management (SCM) 83, 184
support 67
support cases 220-222
System Notes 78, 248
system tables 238

T

tasks
 defining, in project plan 36, 37
tax codes 180
tax features 179, 180
tax types 180
terms 203
test case 38
test plan 38
Text Enhance 122
three-way matching 160
time and materials contract 29
token-based authentication 262
Transfer Orders 203
transitions 246
Trial Balance report 278

U

Undeposited Funds 206
upper management 66
user acceptance testing (UAT) 8, 53, 68
User Event scripts 251, 265
User Interface (UI) centers 12
user stories 27, 28
users, with multiple roles
 managing 73

V

VB matching 160
Vendor Bills 189, 190
Vendor Center dashboard 12, 132, 133
Vendor Credit transaction 193
Vendor Payments 191
Vendor Prepayments (VPPs) 83, 190
vendor rebates 106
Vendor Return Authorizations 192
vendors 90, 95
 addresses 159
 defining, for positive PTP processes 159
 financial considerations 159, 160
 multiple subsidiaries 159
 vendor bill/three-way matching 160
view 229
View and Approve role 163

W

warehouse 67
Warehouse Management System (WMS) 11, 50, 99, 189, 202
waterfall method 23-26
Web to Order (WTO) 103, 108
Wholesale Distribution (WD) 11
workbooks 232
 utilizing, for deeper analysis 232-235
workflows 245
 NetSuite processes, customizing
 with 245-248
Work Order 199



packtpub.com

Subscribe to our online digital library for full access to over 7,000 books and videos, as well as industry leading tools to help you plan your personal development and advance your career. For more information, please visit our website.

Why subscribe?

- Spend less time learning and more time coding with practical eBooks and Videos from over 4,000 industry professionals
- Improve your learning with Skill Plans built especially for you
- Get a free eBook or video every month
- Fully searchable for easy access to vital information
- Copy and paste, print, and bookmark content

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at packtpub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at customercare@packtpub.com for more details.

At www.packtpub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on Packt books and eBooks.

Other Books You May Enjoy

If you enjoyed this book, you may be interested in these other books by Packt:

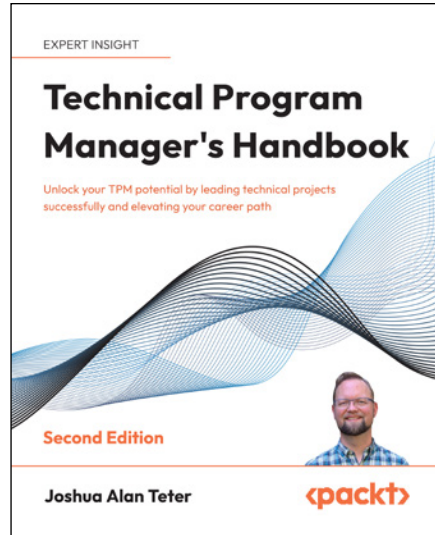


The AI Value Playbook

Lisa Weaver-Lambert

ISBN: 978-1-83546-175-4

- Fundamentals of AI concepts and the tech stack
- How AI works with real-world practical applications
- How to integrate into your company's overall strategy
- How to incorporate generative AI in your processes
- How to drive value with sector-wide examples
- How to organize an AI-driven operating model
- How to use AI for competitive advantage
- The dos and don'ts of AI application



Technical Program Manager's Handbook

Joshua Alan Teter

ISBN: 978-1-83620-047-5

- Uncover the critical importance of the TPM role in the tech industry
- Understand and leverage the unique aspects of the TPM role
- Discover what makes a successful TPM through real-world case studies
- Master project management with advanced technical skills and AI tools
- Apply EI to enhance leadership and team management
- Explore careers and paths for TPMs in the Big Five tech companies

Packt is searching for authors like you

If you're interested in becoming an author for Packt, please visit authors.packtpub.com and apply today. We have worked with thousands of developers and tech professionals, just like you, to help them share their insight with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Share Your Thoughts

Now you've finished *NetSuite for Consultants*, we'd love to hear your thoughts! If you purchased the book from Amazon, please [click here](#) to go straight to the Amazon review page for this book and share your feedback or leave a review on the site that you purchased it from.

Your review is important to us and the tech community and will help us make sure we're delivering excellent quality content..

