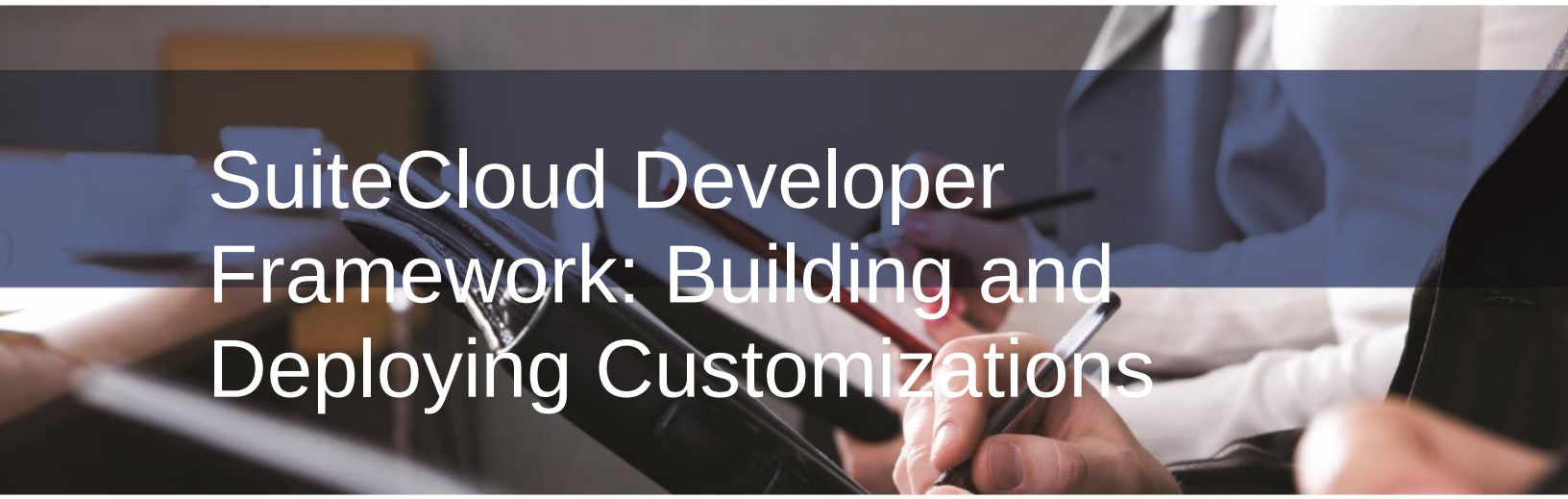
ORACLE
NETSUITE



SuiteCloud Developer Framework: Building and Deploying Customizations

Course Guide

This printing: August 2021.

Oracle NetSuite
2300 Oracle Way
Austin, TX 78741

TABLE OF CONTENTS

Table of Contents ..... **i**

Course Introduction **1**

01: Adjust NetSuite Preferences 1

Exercise Solutions 2

01: Adjust NetSuite Preferences 2

SDF Introduction **5**

EXERCISE 01: Enable Features 5

EXERCISE 02: Set SDF Preferences 5

EXERCISE 03: Download XML 6

Exercise Solutions 7

EXERCISE 01: Enable Features 7

EXERCISE 02: Set SDF Preferences 7

EXERCISE 03: Download XML 9

Getting Started with Account Customization **11**

EXERCISE 01: Create a New Account Customization Project 11

EXERCISE 02: Enable Features 15

Exercise Solutions 17

EXERCISE 01: Create a New Account Customization Project 17

EXERCISE 02: Enable Features 17

Creating Objects with XML **19**

EXERCISE 01: Create Custom Record 19

EXERCISE 02: Add Custom Fields 23

EXERCISE 03: Add Links 27

Exercise Solutions 29

EXERCISE 01: Create Custom Record 29

EXERCISE 02: Add Custom Fields 29

EXERCISE 03: Add Links 29

Working with Reference and Manifest **31**

EXERCISE 01: Create List, Record & Field 31

EXERCISE 02: Edit Manifest 36

EXERCISE 03: Create OS-Browser Compatibility Record 38

Exercise Solutions 44

EXERCISE 01: Create List, Record & Field 44

EXERCISE 02: Edit Manifest 44
Importing Bundles and Objects from NetSuite 45
EXERCISE 01: Import Bootcamp Bundle 45
EXERCISE 02: Inspect and Import Saved Search 47
EXERCISE 03: Compare Local and Account Files 47
EXERCISE SOLUTIONS 49
EXERCISE 1: Import Bootcamp Bundle 49
EXERCISE 02: Inspect and Import Saved Search 49
EXERCISE 03: Compare Local and Account Files 49
Creating Custom Objects in Account 51
EXERCISE 01: Customize a Transaction Form 51
EXERCISE 02: Create a Transaction Body Field 53
EXERCISE 03: Delete a Custom Object 54
EXERCISE SOLUTIONS 55
EXERCISE 01: Customize a Transaction Form 55
EXERCISE 02: Create a Transaction Body Field 55
EXERCISE 03: Delete a Custom Object 55
Developing SuiteApps 57
EXERCISE 1: Create a New SuiteApp Project 57
EXERCISE 2: Reference an Object from another SuiteApp 59
EXERCISE SOLUTIONS 60
EXERCISE 1: Create a New SuiteApp Project 60
EXERCISE 2: Reference an Object from another SuiteApp 60
Managing SuiteScript 61
EXERCISE 01: Create a SuiteScript File 61
EXERCISE 02: Define Script Record and Script Deployment 62
EXERCISE 03: Deploy and Test 63
EXERCISE SOLUTIONS 64
EXERCISE 01: Create a SuiteScript File 64
EXERCISE 02: Define Script Record and Script Deployment 64
EXERCISE 03: Deploy and Test 64
Publishing and Managing SDF Projects 65
EXERCISE 01: Manage Deployment Files 65
EXERCISE 02: Apply Content Protection 66
EXERCISE 03: Bundle a SuiteApp Project 68

EXERCISE SOLUTIONS 70
EXERCISE 01: Manage Deployment Files 70
EXERCISE 02: Apply Content Protection 70
Using the Command Line Interface 71
EXERCISE 01: Set Up SDF CLI 71
EXERCISE 02: Create an Account Customization Project 72
EXERCISE 03: Upload Custom Object 73
Appendix A | Generic IDs 75

COURSE INTRODUCTION

Student Exercises	
01	Adjusting NetSuite Preferences

01: Adjust NetSuite Preferences

Scenario: In this brief exercise, adjust how web pages in NetSuite display the date, time zone, and default language, and SuiteCloud Development Framework preferences.

Note: Course materials assume English (U.S.) as the language, but English (International) is also available.

1. Navigate to **Home > Set Preferences**
2. On the General subtab, update or verify the preferences in the table below.

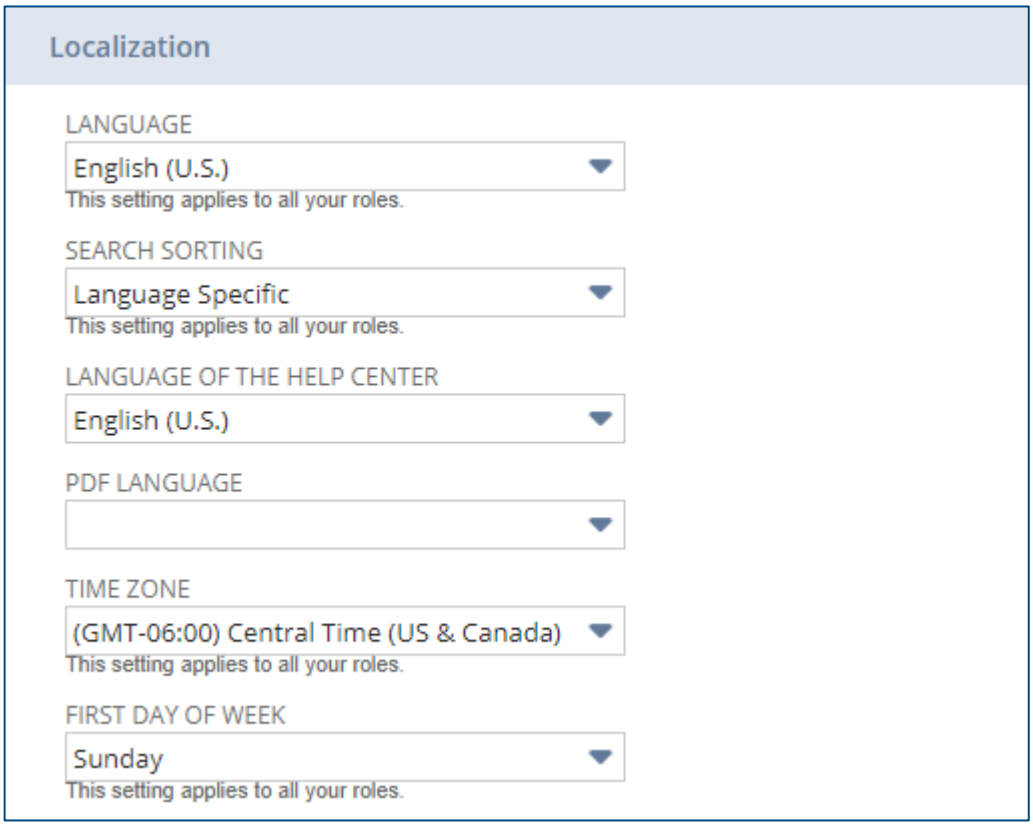
Localization Section	
LANGUAGE	Choose English (U.S.) or English (International).
TIME ZONE	Select the current time zone.
Formatting Section	
DATE FORMAT	Update the desired date format.
Defaults Section	
Show Internal IDs	Allows viewing internal field IDs in NetSuite.

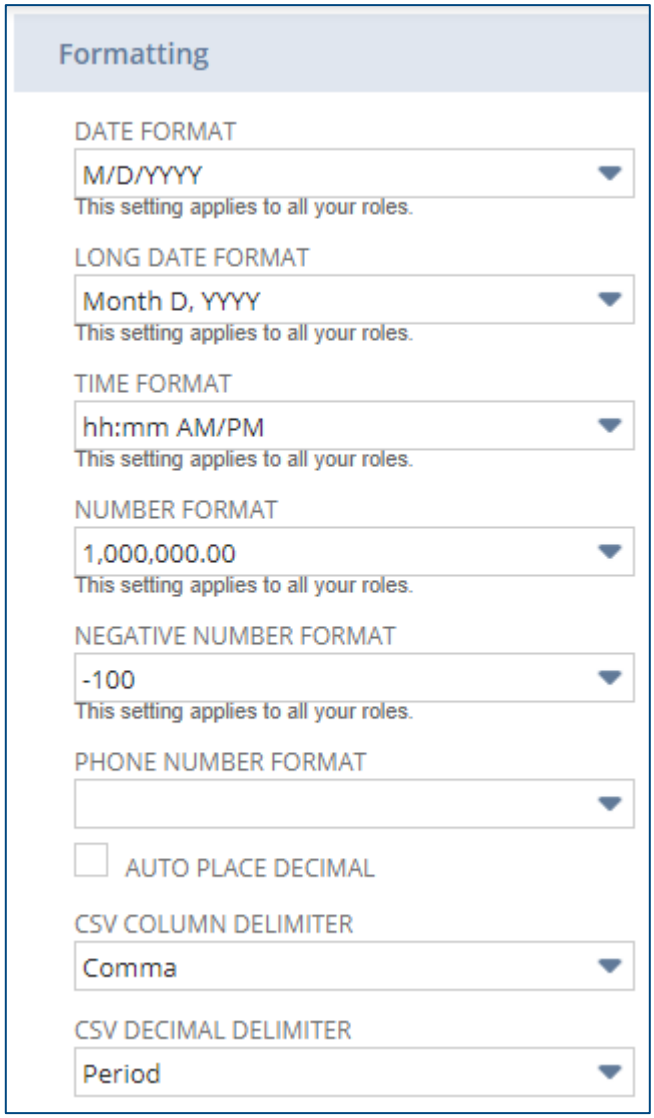
3. On the **Analytics** subtab, check the Show List When Only One Result checkbox (if not already checked). This displays a list when there is a single record rather than opening the record in View mode.
4. Review and set other preferences, as desired.
5. Save any changes when finished.

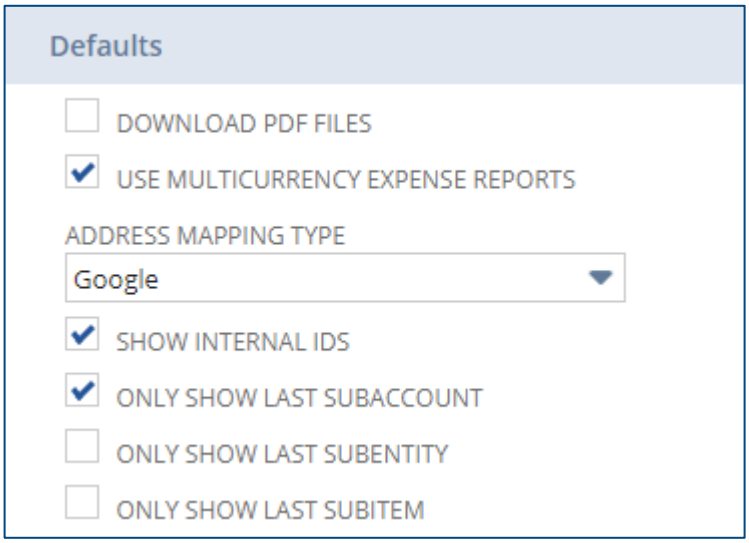
Exercise Solutions

01: Adjust NetSuite Preferences

2

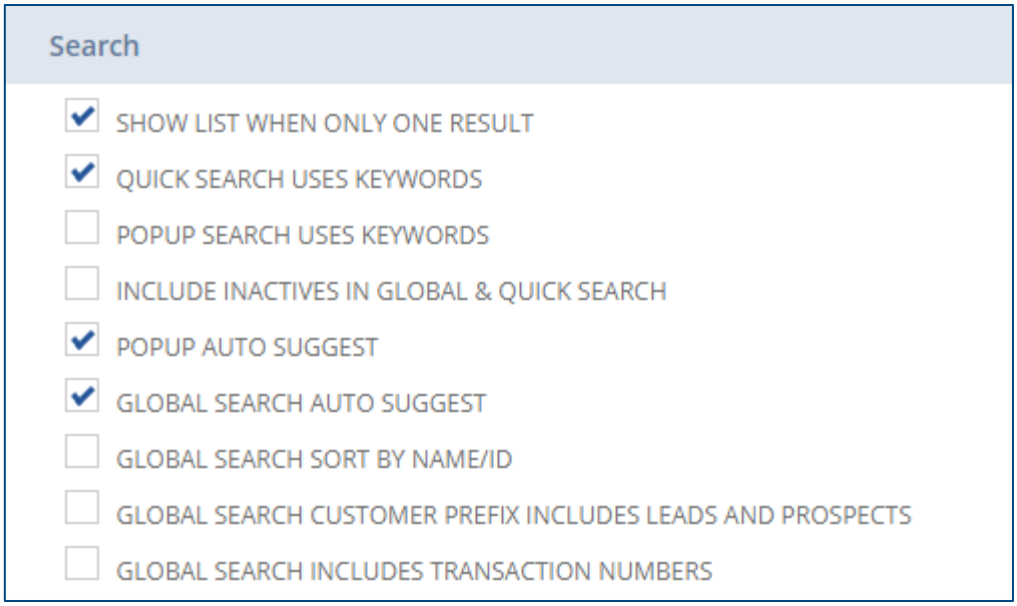






ORACLE NETSUITE

3


Search
SHOW LIST WHEN ONLY ONE RESULT
QUICK SEARCH USES KEYWORDS
POPUP SEARCH USES KEYWORDS
INCLUDE INACTIVES IN GLOBAL & QUICK SEARCH
POPUP AUTO SUGGEST
GLOBAL SEARCH AUTO SUGGEST
GLOBAL SEARCH SORT BY NAME/ID
GLOBAL SEARCH CUSTOMER PREFIX INCLUDES LEADS AND PROSPECTS
GLOBAL SEARCH INCLUDES TRANSACTION NUMBERS

SDF INTRODUCTION

Student Exercises	
01	Enable Features
02	Set SDF Preferences
03	Download XML

EXERCISE 01: Enable Features

Scenario: In this hands-on exercise, verify that SuiteCloud Development Framework (SDF) is enabled in your account.

Enable SuiteCloud Development Framework

1. To enable SDF, navigate to **Setup > Company > Enable Features.**
2. Click the **SuiteCloud** subtab.
3. Locate SuiteCloud Development Framework and ensure it is enabled.

Note: SDF should already be enabled in the NetSuite training account, but the above steps may need to be repeated in your organization's account.

EXERCISE 02: Set SDF Preferences

Scenario: After enabling SDF, APP ID and SHOW ID FIELD will be available in the account. The APP ID field appears on custom objects supported by SDF. Use this to identify objects that can be customized in the SuiteCloud IDE. The SHOW ID FIELD allows adding, editing, and viewing a custom script ID for sublists supported by SDF.

View a Custom List

1. To view a custom list, navigate to Customization > List, Records & Fields > Lists > New.
2. Take a screenshot of the Custom List page.

Enable Preferences

3. To enable preferences, navigate to **Home > Set Preferences**.

4. On the **General** subtab, locate the SuiteCloud Development Framework section.
5. Check SHOW APP ID and SHOW ID FIELD ON SUBLISTS.
6. Click **Save**.

Verify APP ID and Show ID Field on Sublists

7. To verify APP ID and SHOW ID field on sublists, navigate to **Customization** > **List, Records & Fields** > **Lists** > **New**.
8. Compare your screenshot with the current page.
9. If APP ID is enabled successfully, Publisher ID and App ID will appear as choices under the Name field. With the SHOW ID FIELD enabled, you can add an ID before the value under **Values** subtab.
10. Uncheck SHOW ID FIELD ON SUBLISTS in Preferences.

EXERCISE 03: Download XML

Scenario: The purpose of this exercise is to identify the use of XML in NetSuite. Download the XML file from the interface and inspect tags associated with NetSuite objects. XML representations of custom objects are enclosed in "< >" and are usually followed by their value.

Identify XML Values

1. To identify XML values, navigate to Customization > List, Records & Fields > Lists.
2. Select any custom list.
3. Click the Actions dropdown and download the XML file.
4. Open the file with a preferred text editor.
5. Identify the value for the objects in the table below.

Field	Value
CUSTOM LIST SCRIPT ID	
NAME	
IS ORDERED	

Exercise Solutions

EXERCISE 01: Enable Features

3

SuiteCloud Development Framework

SUITECLOUD DEVELOPMENT FRAMEWORK

CREATE SUITECLOUD PROJECTS CONTAINING CUSTOMIZATIONS FOR INTERNAL USE WITHIN YOUR ORGANIZATION OR FOR COMMERCIAL DISTRIBUTION. BY ENABLING THIS FEATURE, YOU AGREE TO THE SUITECLOUD TERMS OF SERVICE.

COPY TO ACCOUNT (BETA)

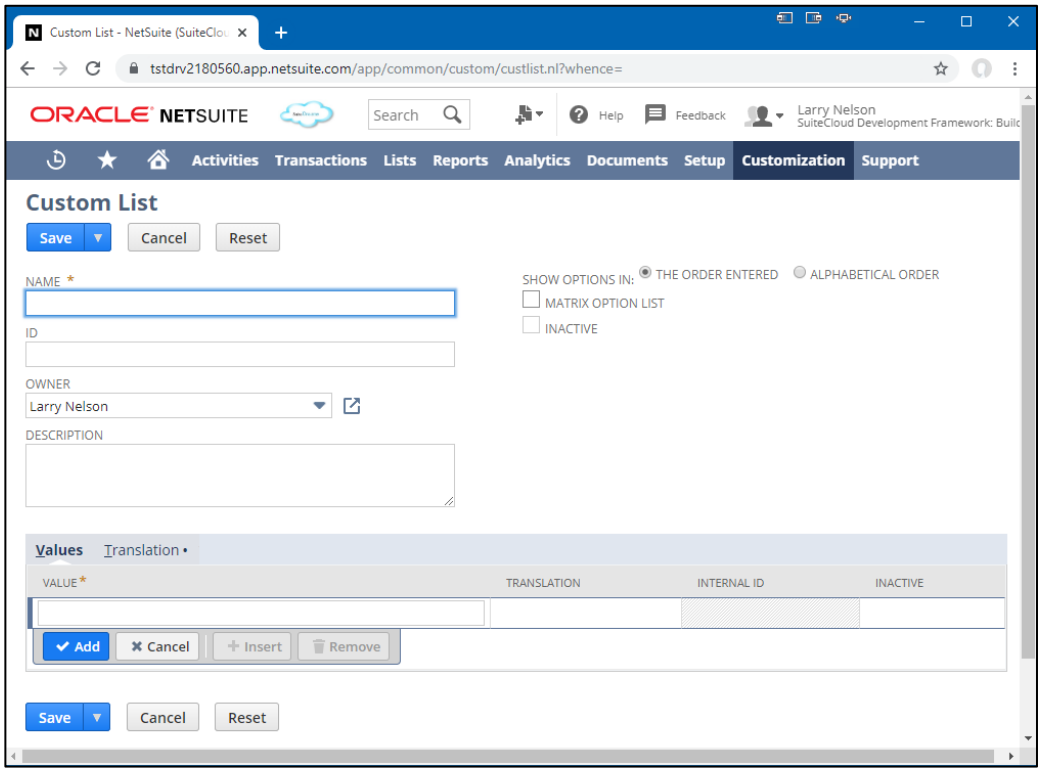
COPY CUSTOMIZATIONS FROM OTHER ACCOUNTS INTO THIS ACCOUNT. BY ENABLING THIS FEATURE, YOU AGREE TO THE SUITECLOUD TERMS OF SERVICE.

SUITEAPP CONTROL CENTER (BETA)

DISTRIBUTE SUITEAPPS TO CUSTOMERS. BY ENABLING THIS FEATURE, YOU AGREE TO THE SUITECLOUD TERMS OF SERVICE.

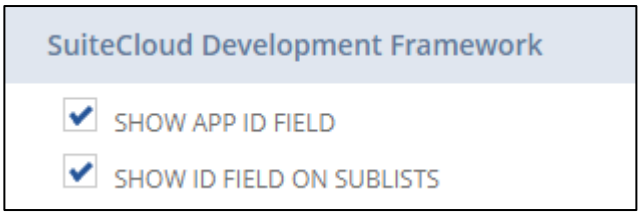
EXERCISE 02: Set SDF Preferences

2

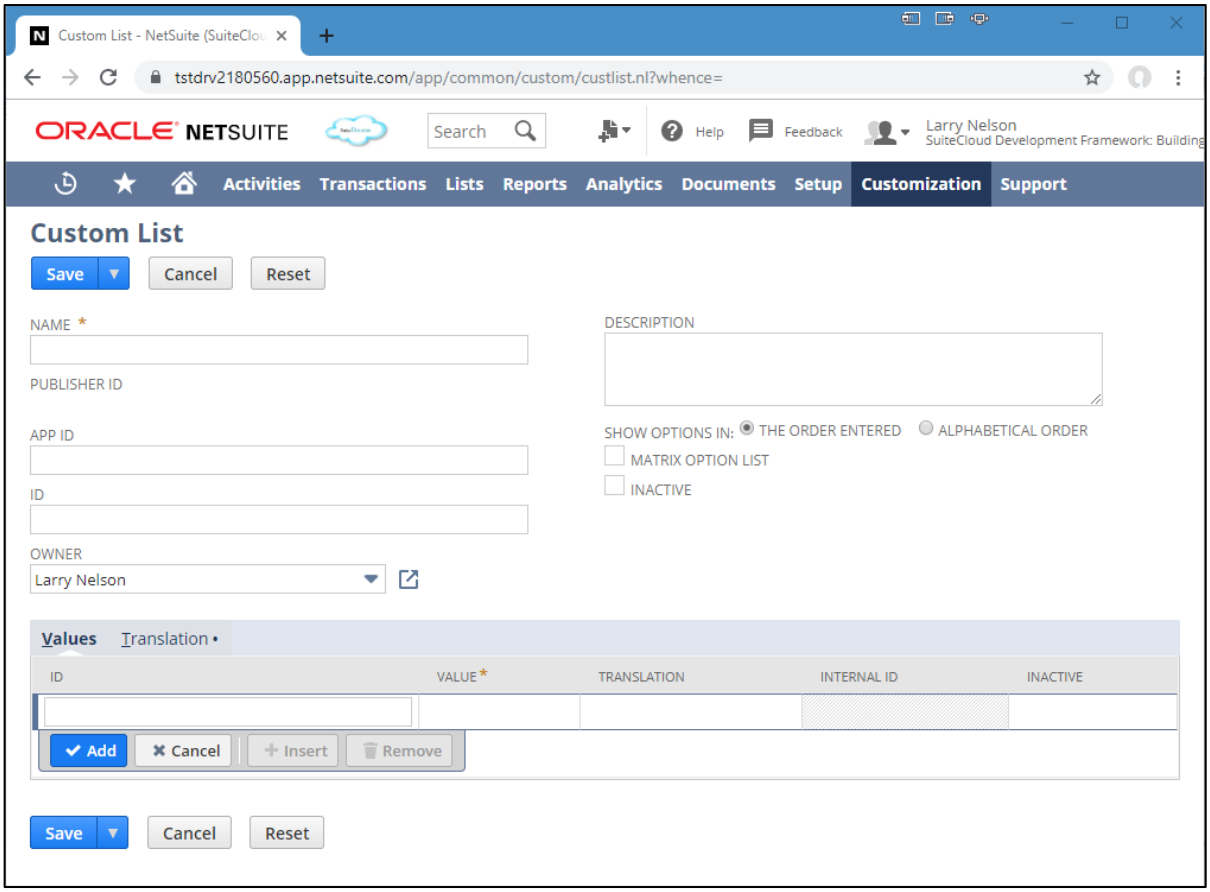


ORACLE NETSUITE

5

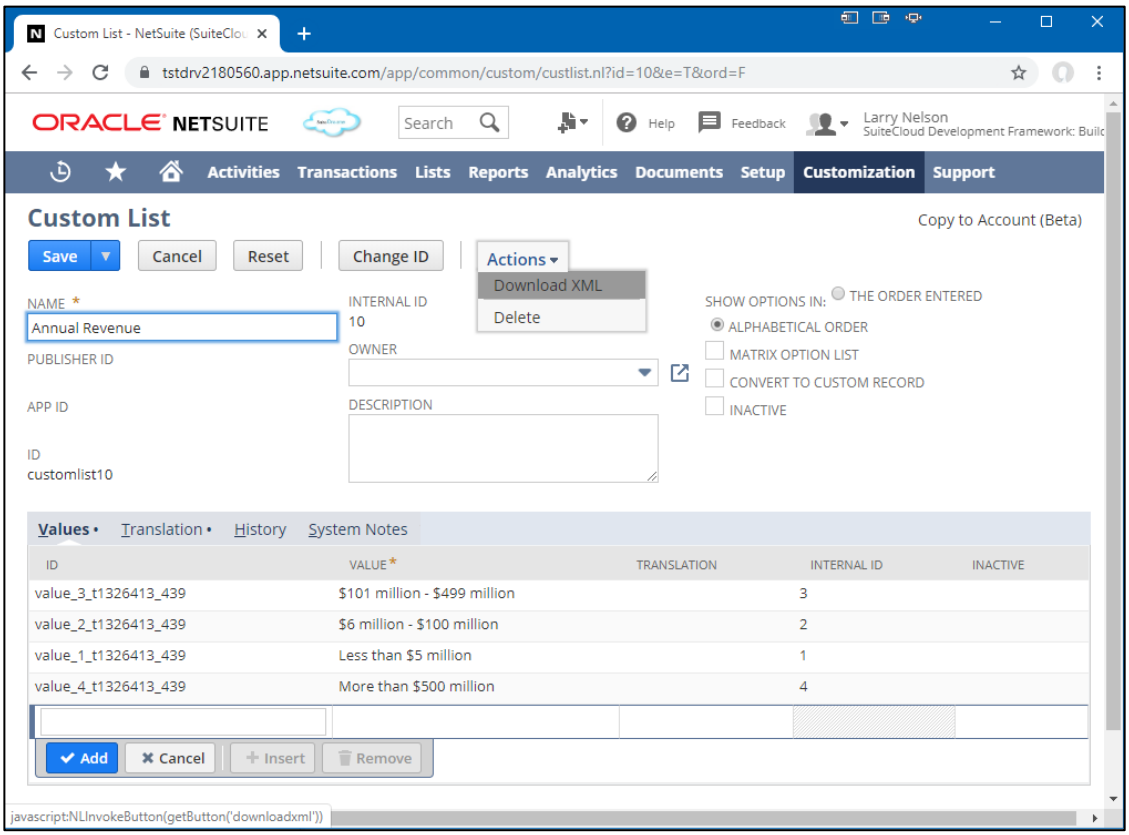

SuiteCloud Development Framework
SHOW APP ID FIELD
SHOW ID FIELD ON SUBLISTS

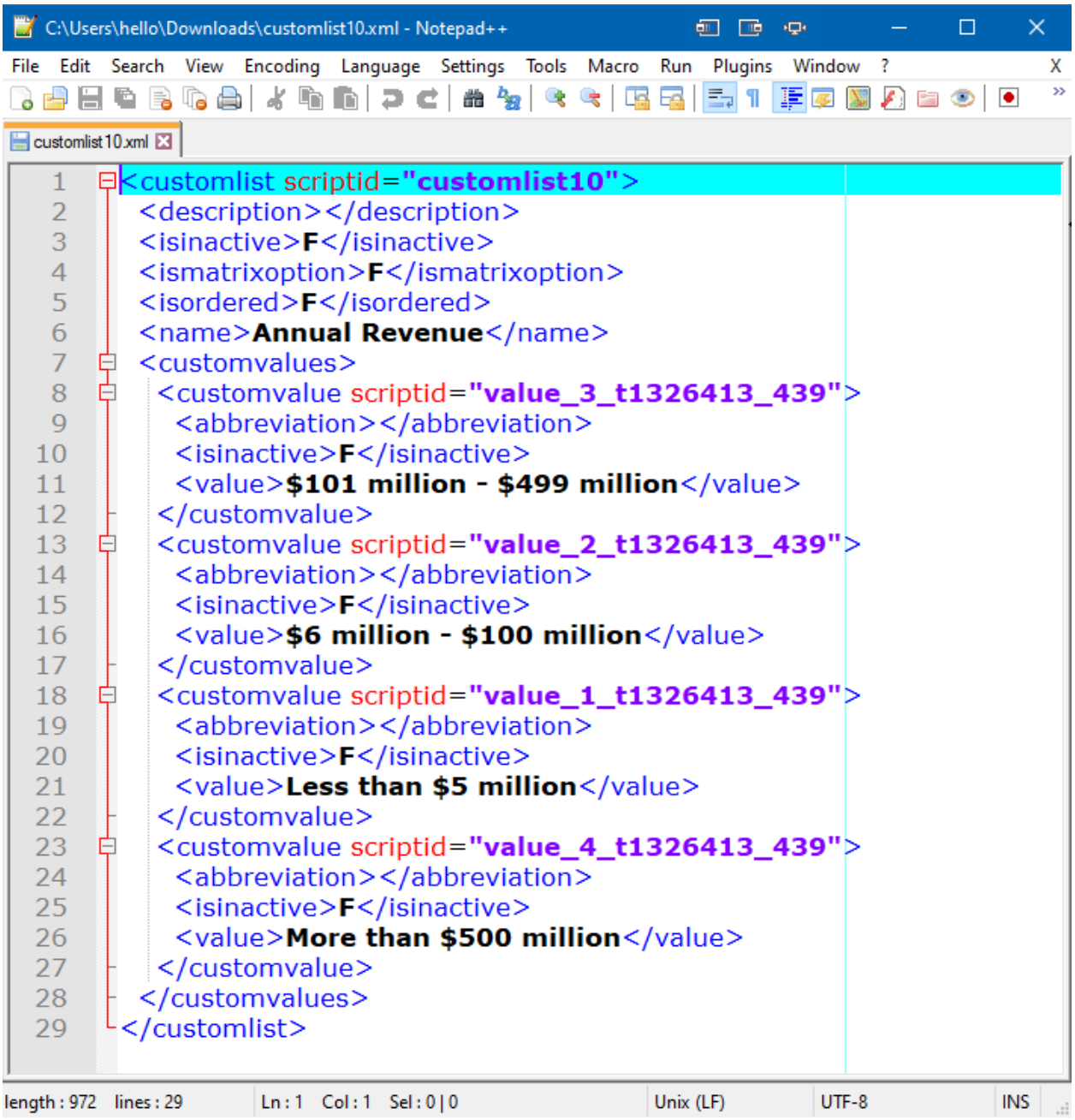
8


Custom List - NetSuite (SuiteClou
tstdrv2180560.app.netsuite.com/app/common/custom/custlist.nl?whence=
ORACLE NETSUITE
Search
Help
Feedback
Larry Nelson
SuiteCloud Development Framework: Building
Activities
Transactions
Lists
Reports
Analytics
Documents
Setup
Customization
Support
Custom List
Save
Cancel
Reset
NAME *
PUBLISHER ID
APP ID
ID
OWNER
Larry Nelson
DESCRIPTION
SHOW OPTIONS IN:
THE ORDER ENTERED
ALPHABETICAL ORDER
MATRIX OPTION LIST
INACTIVE
Values
Translation
ID
VALUE *
TRANSLATION
INTERNAL ID
INACTIVE
Add
Cancel
Insert
Remove
Save
Cancel
Reset

EXERCISE 03: Download XML

5





Field	Value
CUSTOM LIST SCRIPT ID	customlist10
NAME	Annual Revenue
IS ORDERED	F

GETTING STARTED WITH ACCOUNT CUSTOMIZATION

Student Exercises	
01	Create a New Account Customization Project
02	Enable Features

EXERCISE 01: Create a New Account Customization Project

Scenario: You are about to start using SDF for your NetSuite account. Since you are creating an SDF project for your company, you will be creating an account customization project. Account customization provides the flexibility to customize your NetSuite environment using XML representations. In this exercise, create an account customization project and inspect its properties.

Authenticate Master Password

1. In Eclipse, select **NetSuite** > **Master Password** > **Authenticate Master Password**.
2. Enter the password and click **OK**.

You must authenticate the master password each time you launch Eclipse.

Create an Account Customization Project

3. To create an Account Customization (AC) project, in Eclipse, go to **File** > **New** > **SuiteCloud Project**.
4. An option to select the SDF project type appears. Choose **Account Customization** and click **Next**.
5. Set the following properties:
   - **Project Name**: Enable Features
   - **SuiteScript Version**: 2.0
6. Click **Finish**. SuiteCloud IDE generates folders and a manifest file for the SDF project.
7. Use the **NS Explorer** tab to expand the project to view the project structure.

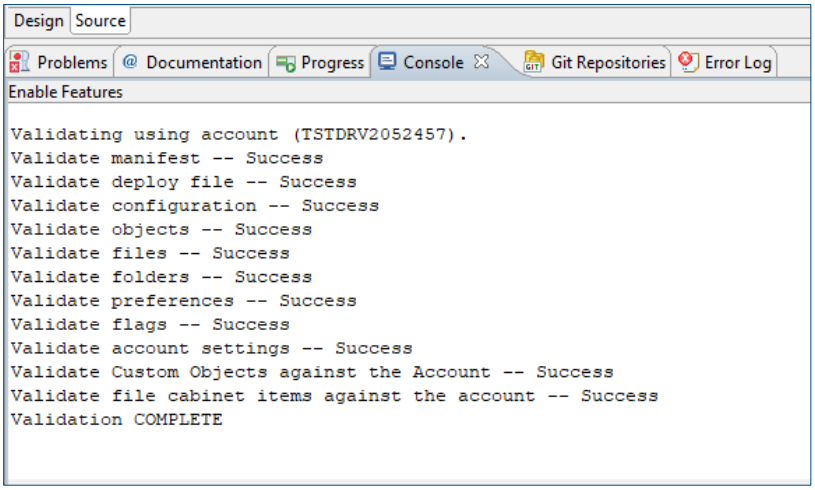
Inspect Elements

8. To inspect elements, from the **Enable Features** project, open **manifest.xml**.
9. Observe the structure of the manifest.xml. The manifest file contains the project type and framework version. Dependency references are added to this file during development.
10. Open the **deploy.xml** file.
11. Observe the structure of the deploy.xml file. You will define the path of objects in the deploy.xml file. It is best practice to ensure that the folder name and location are the same as the deployment path.
12. Right-click **Objects** and select **New** > **Custom Object File**.
13. Select **List** as the Type.
14. Enter **customlist_foods.xml** as the file name.
15. Click **Finish**. You can safely ignore any errors for now.
16. Expand **Objects**. Note that an XML file is generated under the Objects folder. All custom objects will be managed in this folder.
17. Right-click the **customlist_foods.xml** file and click **Delete**.

Validate and Deploy the Project

18. Click **NetSuite** > **Manage Accounts**.
19. Click **Add**.
20. Enter your training account email and password and click **Next >**.
21. Select your training account and click **Finish**.
22. Click **Close**.
23. To validate and deploy the project, from the **NS Explorer** tab, right-click the Enable Features project.
24. Navigate to **NetSuite** > **Validate Project Against Account**.
25. Select the account and role associated with the demo account.

26 Click **Validate**. SuiteCloud IDE validates the project against the selected account. (View the selected account in the **Console** log, as shown below.) Since you are working in a demo account, the account ID begins with "TSTDRV."



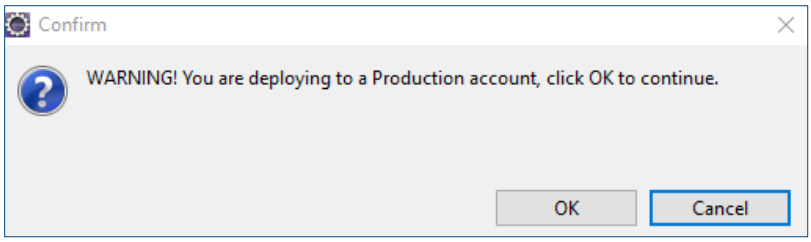
27 After successful validation, deploy the Enable Features project by right-clicking the Enable Features project from the NS Explorer subtab.

28 Go to **NetSuite** > **Deploy** or **NetSuite** > **Deploy to Account.**

29 Provide information for the Account and Role.

30 Click **OK** and **Deploy**.

31 A warning dialog opens. Click **OK**.



32 Observe the Console logs and verify that installation is COMPLETE. The log should appear as follows:

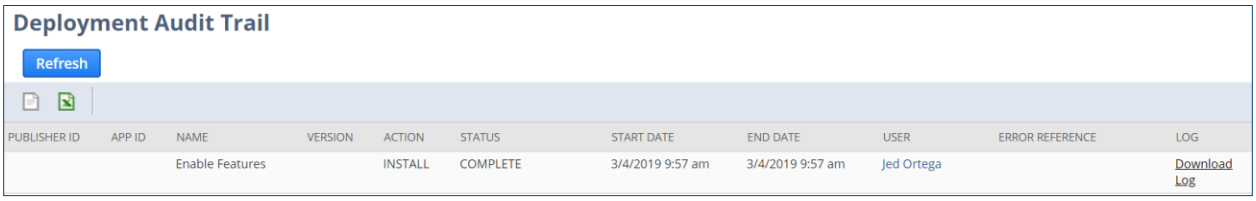
```
[DATEOFDEPLOYMENT] [CURRENT TIME] (PST) Installation started
Info -- Account [(PRODUCTION) SuiteCloud Demo Account]
Info -- Account Customization Project [Enable Features]
Info -- Framework Version [1.0]
Validate manifest -- Success
Validate deploy file -- Success
Validate configuration -- Success
Validate objects -- Success
```

```
Validate files -- Success
Validate folders -- Success
Validate preferences -- Success
Validate flags -- Success
Validate account settings -- Success
Validate Custom Objects against the Account -- Success
Validate file cabinet items against the account -- Success
Begin deployment
2019-03-04 09:57:51 (PST) Installation COMPLETE (0 minutes 1 second)
```

Deployment Audit Trail

33 To verify deployment, log in to NetSuite using the course demo account.

34 In NetSuite, navigate to **Customization** > **SuiteCloud Development** > **Deployment Audit Trail**. The Deployment Audit Trail should appear as shown below:



35 Click **Download Log**. This downloads the same log shown in the Eclipse Console log.

Error in validation and deployment will be added to the deployment log. Use the **Deployment Audit Trail** for team management, as it is easier to collaborate on error messages without checking the IDE of each member.

EXERCISE 02: Enable Features

Scenario: Your company would like to establish a new performance management system, Kudos, which allows coworkers to praise other employees. By default, Kudos is disabled on the account. Use SDF to enable features that do not require a Terms of Service user agreement.

Get Enabled Features in SDF

1. To retrieve enabled features in SDF, in Eclipse, locate the AccountConfiguration folder under the Enable Features project.
2. Right-click the AccountConfiguration folder and navigate to **NetSuite** > **Import Configuration**.
3. Select the demo account and choose the Administrator role.
4. Click **Get Configuration List**.
5. Enable Features appears under the configuration. Check **Enable Features**.
6. Click **OK**.

> **Note:** If imported correctly, features.xml appear under the AccountConfiguration folder.

Enable Kudos

7. To enable Kudos, open the features.xml file.
8. Locate Kudos.
9. Change the status for Kudos from DISABLED to ENABLED. The Kudos feature section should appear as follows:

```
<feature label="Kudos">
   <id>KUDOS</id>
   <status>ENABLED</status>
</feature>
```

10. Validate and deploy.

Verify

11. To verify, log in to NetSuite.com.

12 Navigate to **Setup** > **Company** > **Enable Features** > **Employees** > **Performance Management** > **Kudos**.

13 Click **Setup** > **Performance Management** > **Kudos** > **New** to explore the newly enabled feature.

Exercise Solutions

EXERCISE 01: Create a New Account Customization Project

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Enable Features Exercise 01 Solution.zip**

Import to Eclipse:

2. Right click in **NS Explorer** > **Import...**
3. **General** > **Existing Projects into Workspace**
4. **Next >**
5. **Select archive file:**
6. Select the downloaded file from Step 1.
7. **Select project in Projects:**
8. **Click Finish.**

EXERCISE 02: Enable Features

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Enable Features Exercise 02 Solution.zip**
2. Import to Eclipse.

CREATING OBJECTS WITH XML

Student Exercises	
01	Create Custom Record
02	Add Custom Fields
03	Add Links

EXERCISE 01: Create Custom Record

Scenario: Create custom records in SDF by having the NetSuite account generate XML objects. As a new SDF developer, create a custom record by manually defining its XML objects. Identifying the XML representations enables customized objects without the NetSuite user interface (UI). In this exercise, practice creating a custom record for food recipes.

Create a New Account Customization Project

1. To create a new Account Customization project, in Eclipse, navigate to **File** > **New** > **SuiteCloud Project**.
2. Choose **Account Customization** and click **Next**.
3. Set the following properties:
4. **Project Name**: Food Recipe
5. SuiteScript Version: 2.0
6. Click **Finish**.

Create a New Custom Object File

7. To create a new custom object file, select the Food Recipe project under the **NS Explorer** subtab.
8. Right-click the project and navigate to **New** > **Custom Object File**. A new Custom Object File opens.
9. Set the properties according to the table below.

Property	Value
Type	Record Type
Parent Folder	Food Recipe/Objects
Filename	customrecord_sdr_foodrecipe.xml

10 Click **Finish**. An XML file generates within the Objects folder.

11 Open customrecord_sdr_foodrecipe.xml.

12 Locate the XML representation for <recordname> and set it to **Food Recipe**. The <recordname> representation should appear as follows:

```
<recordname>Food Recipe</recordname>
```

Customize a Field

13 To customize a field, locate the XML representation for a custom field. The XML block should appear as follows:

```
<customrecordcustomfield scriptid="custrecord_">
      <accesslevel>2</accesslevel>
      <description></description>
      <displaytype>NORMAL</displaytype>
      <fieldtype>TEXT</fieldtype>
      <label></label>
      <searchlevel>2</searchlevel>
      <storevalue>T</storevalue>
  </customrecordcustomfield>
```

14 Set the properties according to the table below.

Property	Value
scriptid=	custrecord_sdr_dish
<fieldtype>	TEXT
<label>	Dish

15 **Save** the file.

16 In the **NS Explorer** subtab, right-click Food Recipe and navigate to **NetSuite** > **Validate Project Against Account**.

17 The following validation error appears:

```
Validation failed.

An error occurred during custom object validation.
(customrecord_sdr_foodrecipe)
Details: When the SuiteCloud project contains a customrecordtype, the manifest
must define the CUSTOMRECORDS feature as required
File: ~/Objects/customrecord_sdr_foodrecipe.xml
```

18 Under the Food Recipe folder, right-click the manifest.xml file and navigate to **NetSuite** > **Add Dependency References to Manifest**.

Note: The Add Dependency References to Manifest action automatically defines feature dependencies for the account. NetSuite features (such as custom records) require manually referencing the dependency in the SDF project so that validation and deployment succeeds.

19 Deploy to the account.

Verify Deployment

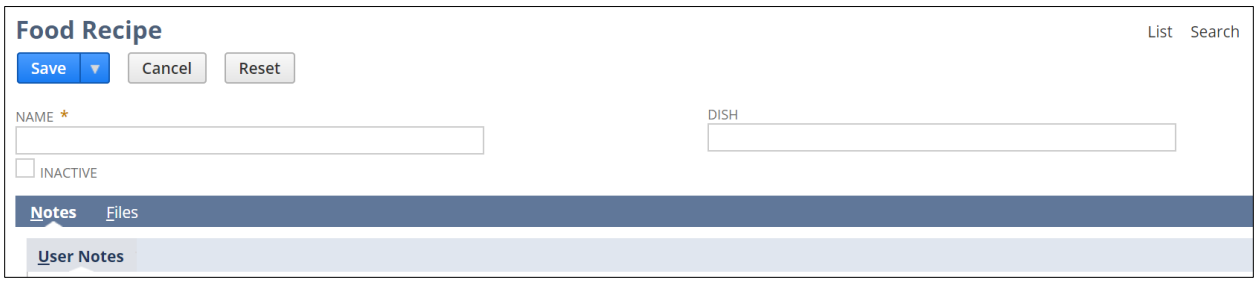
20 To verify deployment, log in to NetSuite using the course demo account.

21 In NetSuite, navigate to **Customization** > **Lists, Records, & Fields** > **Record Types**.

22 Click **New Record** for Food Recipe.

Note: If you cannot locate Food Recipe, check the status of the Deployment Audit Trail and ensure it indicates COMPLETE. Return to the IDE to fix the error and re-try account deployment.

23 Verify that the DISH field is available from the record and click **Cancel**.



Remove Other Fields

24 To remove fields, in Eclipse, open the Edit customrecord_sdr_foodrecipe.xml file.

25 Modify the properties according to the table below.

Property	Value
<allowattachments>	F
<enablesystemnotes>	F
<includename>	F
<shownotes>	F

The modification should appear as follows:

```
<allowattachments>F</allowattachments>
<allowinlinedetaching>T</allowinlinedetaching>
<allowquickadd>T</allowquickadd>
<enabledle>T</enabledle>
<enablekeywords>T</enablekeywords>
<enableoptimisticlocking>T</enableoptimisticlocking>
<enablesystemnotes>F</enablesystemnotes>
<iconbuiltin>T</iconbuiltin>
<includeinsearchmenu>T</includeinsearchmenu>
<includename>F</includename>
<shownotes>F</shownotes>
<recordname>Food Recipe</recordname>
```

Note: This modification removes the NAME field, **Notes** subtab, and **Files** subtab.

26 **Save** the file.

Set a Mandatory Field

27 To set a mandatory field, in the <custrecord_sdr_dish> field block, add an <ismandatory> tag.

28 Set <ismandatory to> T. The tag should appear as follows:

```
<customrecordcustomfield scriptid="custrecord_sdr_dish">
          <ismandatory>T</ismandatory>
```

Note: <ismandatory> makes the field mandatory or required for the user, which also requires that you provide a Boolean value (F or T).

29 **Save** the file.

30 Deploy.

Verify in NetSuite

31 To verify, in NetSuite, navigate to Customization > Lists, Records, & Fields > Record Types.

32 Click New Record for Food Recipe.

33 Verify that the DISH field is now mandatory, indicated by an asterisk (*).

EXERCISE 02: Add Custom Fields

Scenario: After creating a custom record, add custom fields by editing the custom record XML file. In this exercise, add fields to collect requirements for a Food Recipe record. Create the following fields:

Prep Time (Mins)

Cook Time (Mins)

Serves

Ingredients

Method

Prep Time (Mins) Field

1 To create the Prep Time (Mins) field, in Eclipse, expand Food Recipe project.

2 Edit the customrecord_sdr_foodrecipe.xml file.

3 Copy the <custrecord_sdr_dish> XML block, which appears as follows:

```
<customrecordcustomfield scriptid="custrecord_sdr_dish">
        <ismandatory>T</ismandatory>
        <accesslevel>2</accesslevel>
        <description></description>
        <displaytype>NORMAL</displaytype>
        <fieldtype>TEXT</fieldtype>
        <label>Dish</label>
        <searchlevel>2</searchlevel>
        <storevalue>T</storevalue>
 </customrecordcustomfield>
```

4. Paste the XML block below the current block, creating a duplicate.

   Note: Make sure to paste the custom field block inside the `<customrecordcustomfields>……</customrecordcustomfields>` tag.

5. Edit the properties of the duplicate custom field block using the table below to create the Prep Time (Mins) field.

Property	Value
scriptid=	custrecord_sdr_preptime
<ismandatory>	F
<fieldtype>	INTEGER
<label>	Prep Time (Mins)

   The code should appear as follows:

```
<customrecordcustomfield scriptid="custrecord_sdr_preptime">
        <ismandatory>F</ismandatory>
            <accesslevel>2</accesslevel>
            <description></description>
            <displaytype>NORMAL</displaytype>
            <fieldtype>INTEGER</fieldtype>
            <label>Prep Time (Mins)</label>
            <searchlevel>2</searchlevel>
            <storevalue>T</storevalue>
 </customrecordcustomfield>
```

6. Save the file.

7. Deploy the record.

8. Log in to NetSuite and ensure the new field appears as it should.

Cook Time (Mins) Field

9. To create the Cook Time (Mins) field, in Eclipse, edit the customrecord_sdr_foodrecipe.xml file from the Food Recipe project.

10. Create a new field by copying the previous XML block.

11 Edit the properties using the table below to create the Cook Time (Mins) field.

Property	Value
scriptid=	custrecord_sdr_cooktime
<mandatory>	F
<fieldtype>	INTEGER
<label>	Cook Time (Mins)

12 Save the file.

Serves Field

13 To create the Serves field, first repeat step 9 and step 10.

14 Edit the properties using the table below to create the Serves field.

Property	Value
scriptid=	custrecord_sdr_serves
<mandatory>	F
<fieldtype>	INTEGER
<label>	Serves

15 Save the file.

Ingredients Field

16 To create the Ingredients field, first repeat step 9 and step 10.

17 Edit the properties using the table below to create the Ingredients field.

Property	Value
scriptid=	custrecord_sdr_ingredients
<Mandatory>	T
<fieldtype>	Long Text/CLOBTEXT
<label>	Ingredients

18 Save the file.

Methods Field

19 To create the Methods field, first repeat step 9 and step 10.

20 Edit the properties using the table below to create the Methods field.

Property	Value
scriptid=	custrecord_sdr_method
<mandatory>	T
<fieldtype>	Long Text/CLOBTEXT
<label>	Method

21 **Save** the file.

22 Deploy the record.

23 Log in to NetSuite and ensure that the Food Recipe record appears with the newly created fields.

Modify Field Attributes

24 To modify attributes, edit the customrecord_sdr_foodrecipe.xml. file from the Food Recipes project.

25 Go to the <custrecord_sdr_serves> field XML block.

26 Add a <minvalue> tag and set the value to **1**.

27 Add a <maxvalue> tag and set the value to **360.**

> **Note**: <minvalue> and <maxvalue> tags allow setting a minimum and maximum value for the field.

28 Add a <defaultvalue> tag and set the value to **1**.

29 **Save** the record. The code should appear as follows:

```
<customrecordcustomfield scriptid="custrecord_sdr_serves">
 <ismandatory>F</ismandatory>
     <accesslevel>2</accesslevel>
     <description></description>
     <displaytype>NORMAL</displaytype>
     <fieldtype>INTEGER</fieldtype>
     <minvalue>1</minvalue>
     <maxvalue>360</maxvalue>
     <defaultvalue>1</defaultvalue>
     <label>Serves</label>
     <searchlevel>2</searchlevel>
     <storevalue>T</storevalue>
 </customrecordcustomfield>
```

30 Deploy the record and verify that the fields appear as they should in NetSuite.

31 Return to the Food Recipe record and input a value in the SERVES field. Attempt to input **0** (under the minimum) and **700** (over the maximum).

EXERCISE 03: Add Links

Scenario: In this exercise, configure and add links to a custom record under **Lists** > **Custom** subtab.

Identify XML Values

1 To identify XML values, expand the Food Recipe project in Eclipse.

2 Edit the customrecord_sdr_foodrecipe.xml file.

3 Locate closing tag (>) for the custom record type.

4 Create links objects in the following format:

```
<links> <link>…</link>…</links>
```

5 Inside the <link> tag, add a <linkcategory> tag.

6 Set the value of the <linkcategory> tag to BASICLISTSCUSTOM.

> **Note**: This adds the custom record to the **Lists** > **Custom** subtab. Use code completion to identify other categories.

7 Add a <linktasktype> tag and set and set the value to LIST.

8 **Save** the file.

9 Deploy the file to the account.

10 In NetSuite, navigate to **List** > **Custom**.

11 Verify that **Food Recipe** > **New** is now available.

Exercise Solutions

EXERCISE 01: Create Custom Record

Download from File Cabinet:

1. **SDF Files > Exercise Solutions > Food Recipe Exercise 02 Solution.zip**
2. Import to Eclipse.

EXERCISE 02: Add Custom Fields

Download from File Cabinet:

3. **SDF Files > Exercise Solutions > Food Recipe Exercise 02 Solution.zip**
4. Import to Eclipse.

EXERCISE 03: Add Links

Download from File Cabinet:

1. **SDF Files > Exercise Solutions > Food Recipe Exercise 03 Solution.zip**
2. Import to Eclipse.

WORKING WITH REFERENCE AND MANIFEST

Student Exercises	
01	Create List, Record & Field
02	Edit Manifest
03	Create OS-Browser Compatibility

EXERCISE 01: Create List, Record & Field

Scenario: Your company started providing free coffee every week. Everyone is happy about their free coffee, but the company administrator documents all transactions on a spreadsheet. The administrator then asked the NetSuite administrator to create a more efficient way to track all coffee requests. Since your company uses SDF, you are tasked with creating a custom record with dependencies from different NetSuite objects. In this exercise, create a Coffee Request record and reference lists on a custom field.

Prerequisite: Create an Account Customization project (named “Coffee Request”).

Create a Coffee Request Record

1. To create a Coffee Request record, in Eclipse, right-click **Objects** and navigate to **New** > **Custom Object File**.
2. From the Type field, select Record Type.
3. Enter **customrecord_coffeerequest.xml** as the file name.
4. Click **Finish**.
5. Enter **Coffee Request** as the record name.
6. Set <includename> to F.

7 Use the table below to set the properties for the custom field.

Property	Value
scriptid=	custrecord_sdr_cr_emp
<fieldtype>	SELECT
<selectrecordtype>	-4
<label>	Employee
<isparent>	T

8 **Save** the file.

9 Right-click **manifest** and select **Add Dependency References to Manifest**.

10 Validate the project.

Create a Coffee List

11 To create a Coffee list, in Eclipse, right-click **Objects** and navigate to **New** > **Custom Object File**.

12 Input the values for the list using the properties identified in the table below.

Property	Value
Type	List
Parent Folder	Coffee Request/Objects
Filename	customlist_sdr_coffee.xml

13 Edit the customlist_sdr_coffee.xml file and set its properties using the table below.

Property	Value
<name>	Coffee
<customvalue scriptid>	_sdr_cof_1
<Value>	Short Black

14 Populate list with the set of custom values identified in the tables below.

Property	Value
<customvalue scriptid>	_sdr_cof_2
<value>	Doppio

Property	Value
<customvalue scriptid>	_sdr_cof_3
<value>	Short Macchiato

Property	Value
<customvalue scriptid>	_sdr_cof_4
<value>	Long Macchiato

Property	Value
<customvalue scriptid>	_sdr_cof_5
<value>	Americano

Property	Value
<customvalue scriptid>	_sdr_cof_6
<value>	Latte

Property	Value
<customvalue scriptid>	_sdr_cof_7
<value>	Cappuccino

Property	Value
<customvalue scriptid>	_sdr_cof_8
<value>	Mocha

The code should appear as follows:

```
<customlist scriptid="customlist_sdr_coffee">
<isordered>T</isordered>
<name>Coffee</name>
   <customvalues>
      <customvalue scriptid="_sdr_cof_1">
         <value>Short Black</value>
      </customvalue>
      <customvalue scriptid="_sdr_cof_2">
         <value>Doppio</value>
      </customvalue>
      <customvalue scriptid="_sdr_cof_3">
         <value>Short Macchiato</value>
      </customvalue>
      <customvalue scriptid="_sdr_cof_4">
         <value>Long Macchiato</value>
      </customvalue>
      <customvalue scriptid="_sdr_cof_5">
         <value>Americano</value>
      </customvalue>
      <customvalue scriptid="_sdr_cof_6">
         <value>Latte</value>
      </customvalue>
      <customvalue scriptid="_sdr_cof_7">
         <value>Cappuccino</value>
      </customvalue>
      <customvalue scriptid="_sdr_cof_8">
         <value>Mocha</value>
      </customvalue>
</customvalues>
</customlist>
```

15 **Save** the file.

Reference the Coffee List to a Custom Field

16 To reference the Coffee list to a custom field, edit the customrecord_coffeerequest.xml file in Eclipse.

17 Add a new custom field and use the table below to set its properties.

Property	Value
scriptid=	custrecord_sdr_cr_coflist
<fieldtype>	SELECT
<selectrecordtype>	[scriptid=customlist_sdr_coffee]
<label>	Coffee
<isparent>	F
<ismandatory>	T

18 **Save** the file.

19 Validate and deploy the record.

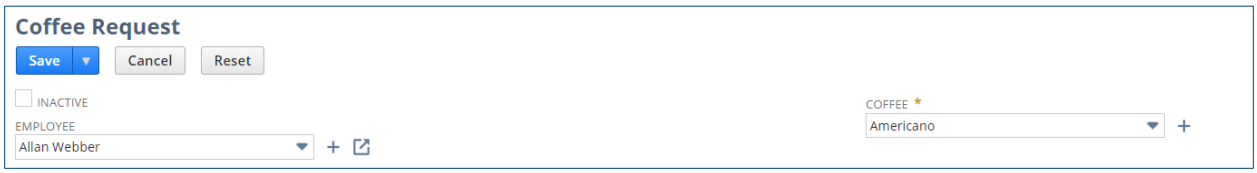
Test the Coffee Request Record

20 Test the Coffee Request record by logging in to NetSuite and navigating to **Customization** > **Lists, Records, & Fields** > **Record Types**

21 Click **New Record** for Coffee Request.

22 Select an Employee and a preferred Coffee.

23 **Save** the record.



Coffee Request record

EXERCISE 02: Edit Manifest

Scenario: The NetSuite administrator created a list of extras (e.g., extra espresso shot) for the Coffee Request record. The administrator created the list in the account; now reference the list in the Coffee Request project. In this exercise, create a field and add the custom list on the manifest to reference the account object.

Check Coffee Extras List ID

1. To check the Coffee Extras list ID, log in to NetSuite and navigate to **Customization** > **Lists, Records, & Fields** > **Lists**.
2. Take note of the ID for the Coffee Extras list.

Add a Custom Field

3. To add a custom field, edit the customrecord_coffeerequest.xml file in Eclipse.
4. Add a new custom field with the properties identified in the table below.

Property	Value
scriptid=	custrecord_sdr_coffee_extras
<fieldtype>	MULTISELECT
<selectrecordtype>	[scriptid=customlist_sdr_cr_cofextra]
<label>	Extras
<isparent>	F
<ismandatory>	F

5. **Save the file**. The following error appears:

```
The object referenced in "selectrecordtype" is missing in the project is not
included in the dependencies list.
```

> **Note:** Since the list is not in the project, edit the manifest to reference the account object.

6. Edit the manifest.xml file.
7. Under the XML block for <features> </features>, add <objects> </objects>.

8 Add <object> (Extras List ID) </object> inside the objects. The should appear as follows:

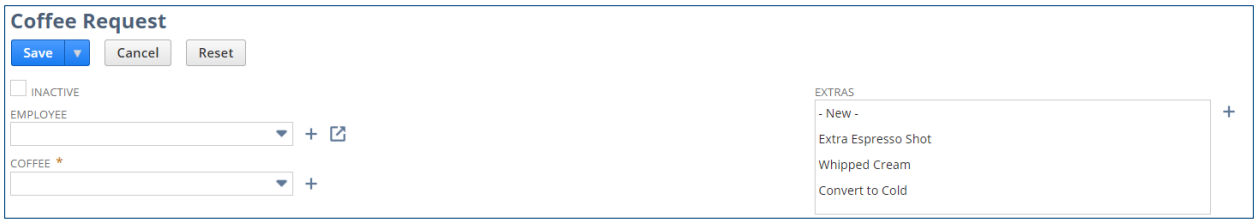
```
<objects>
   <object>customlist_sdr_coffee_xtras</object>
</objects>
```

Note: Adding the ID on manifest allow SDF to check if the object exists on the target account during validation.

9 Validate project and notice that the previous error is resolved.

10 Deploy the record to the account.

11 Log in to NetSuite and verify the changes.



Coffee Request with Extras field

EXERCISE 03: Create OS-Browser Compatibility Record

Scenario: Your company supports employees using various operating systems (OS), but the administrator identifies which operating system and browser employees are to use.

Since not all browsers work with all operating systems, dependent drop-down lists are used on records to help an administrator with this identification task. Dependent drop-downs allow users to make a choice in situations where some options do not make sense when selected together.

The process to create an OS-Browser Compatibility record type is as follows:

- Create an Operating System custom list.
- Create a custom record type containing all valid combinations of operating systems and compatible browsers.
- Create a custom List/Record field that points to the Operating Systems list.
- Create instances of different browser and OS combinations
- Create a custom List/Record field that points to the custom record containing the combinations of OSs and browsers.

Prerequisite: Create an Account Customization project (named “Browser Selection”).

Create a Custom List for Operating Systems

1. To create a custom list for OSs, in Eclipse, right-click **Objects** and navigate to **New** > **Custom Object File**.
2. Select **List** as the Type.
3. Enter **customlist_operating_systems.xml** as the file name.
4. Click **Finish**.

5 Use the table below to provide a value for its corresponding XML representation.

Property	scriptid	Value
<name>		Operating Systems
<customvalue>	windows	Windows
<customvalue>	mac	Mac OS
<customvalue>	linux	Linux

Note: Use Ctrl + Space for code completion.

6 **Save** the file.

Create a Record Type for Compatible Browsers

7 To create a Compatible Browsers record type, create new Custom Object file.

8 Select **Record Type** as the type.

9 Enter **customrecord_compatible_browsers.xml** as the file name and click **Finish**.

10 Remove the <enabledle> tag.

11 Set the following XML representations to false (F):

- Show Notes
- Allow Attachments

12 Enter **Compatible Browsers** as the record name.

Create an Operating System Field for Compatible Browsers

13 To create an Operating System field for compatible browsers, locate the <customrecordcustomfields> and </customrecordcustomfields> tags. Inside these tags define the format for creating the field.

14 Use the table below to set values for the OPERATING SYSTEM field.

Property	Value
scriptid=	custrecord_os_field
<fieldtype>	MULTISELECT
<selectrecordtype>	[scriptid=customlist_operating_systems]
<label>	Operating Systems
<storevalue>	T
<showinlist>	T

15 **Save** the file.

Create Instances of this Record Type

16 Create an instance of each of the following browser/OS combinations:

- Chrome: Windows, Mac OS, Linux
- Safari: Windows, Mac OS
- Internet Explorer: Windows
- Firefox: Windows, Linux

Use the following syntax:

```
<instances>
        <instance scriptid="">
        <custrecord_osfield> </custrecord_osfield>
        <isinactive> </isinactive>
        <name> </name>
        </instance>
</instances>
```

Create a Record Type for Browser Selection

17 To create a Browser Selection record type, create new Custom Object file.

18 Select **Record Type** for Type.

19 Enter **customrecord_employee_osb.xml** as the file name and click **Finish**.

20 Set the following XML representations to false (F):

- Include Name
- Show Notes
- Allow Attachments

21 Enter **Employee OS-Browser** as the record name.

Create Fields for Browser Selection

22 To create fields for browser selection, create an Employee field and use the table below to set its values.

Property	Value
scriptid=	custrecord_employee_osb
<fieldtype>	Select or List/Record
<isparent>	T
<label>	Employee
<parentsubtab>	ENTITYSUPPORT
<selectrecordtype>	-4
<showinlist>	T
<storeinvalue>	T

23 Create an employee Operating System field with values defined in the table below.

Property	Value
scriptid=	custrecord_employee_operating_system
<fieldtype>	Select or List/Record
<label>	Operating System
<selectrecordtype>	
<showinlist>	T
<storeinvalue>	T

24 Create an employee Operating System field with the values defined in the table below.

Property	Value
scriptid=	custrecord_employee_operating_system
<fieldtype>	Select or List/Record
<label>	Operating System
<selectrecordtype>	Operating Systems
<showinlist>	T
<storeinvalue>	T

25 Set values using the table below for the Employee Browser field.

Property	Value
scriptid=	custrecord_employee_browser
<fieldtype>	Select or List/Record
<label>	Browser
<selectrecordtype>	Compatible Browsers
<showinlist>	T
<storeinvalue>	T

26 Add sourcing and filtering to the Employee Browser field using the following syntax:

```
<customfieldfilters>
    <customfieldfilter>
        <fldcomparefield>……</fldcomparefield>
        <fldfilter>…….</fldfilter>
        <fldfiltercomparetype>…….</fldfiltercomparetype>
        </customfieldfilter>
</customfieldfilters>
```

27 Set the values as follows:

- **Compare Field**: Employee Operating System
- **Filter**: Compatible Browser Operating System
- **Compare Type**: EQ

Deploy and Test

28 To deploy and test, first validate and deploy.

29 In NetSuite, navigate to **Customization** > **Lists, Records, & Fields** > **Record Types**.

30 Click **New Record** for the Employee OS-Browser record type.

31 Select **Allan Webber** in the Employee field.

32 Select **Windows** in the Operating System field.

33 Click the Browser drop-down and note the list of browsers.

34 Change the Operating System and select **Mac**.

35 Click the Browser dropdown and select **Safari**.

36 **Save** the record. You are re-directed to the Employee OS-Browser page.

37 Select **Allan Webber** under Employee or perform a global search by entering **emp: Allan**.

38 Click the **Support** subtab to view the Employee OS-Browser sublist.

Exercise Solutions

EXERCISE 01: Create List, Record & Field

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Coffee Request Exercise 01 Solution.zip**
2. Import to Eclipse.

EXERCISE 02: Edit Manifest

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Coffee Request Exercise 02 Solution.zip**
2. Import to Eclipse.

IMPORTING BUNDLES AND OBJECTS FROM NETSUITE

Student Exercises	
01	Import Bootcamp Bundle
02	Inspect and Import Saved Search
03	Compare Local and Account Files

EXERCISE 01: Import Bootcamp Bundle

Scenario: A former developer created a BootCamp Bundle on your NetSuite account and the SDF chose to use SDF to manage the bundle. The BootCamp Bundle records the performance of new hires undergoing training. As the SDF developer, you are tasked with importing the BootCamp bundle and modifying it to fit company training requirements.

Prerequisite: Create a new account customization project and name it as "BootCamp Bundle."

Import Bundle Components

1. To import components of the BootCamp bundle, in Eclipse, select Bootcamp Bundle. Then right-click and navigate to **NetSuite** > **Import Bundle Components**.
2. Select your account and role.
3. Click **List Bundles**.
4. Click the **Bundles** drop-down menu and select **SDF Bootcamp Bundle**.
5. Click **OK**.
6. A confirmation window opens. Click **OK**.
7. Wait for the Import Wizard to finish and click **OK**.
8. Expand the **Objects** folder, which should contain the following files:
   - customlist_sdc_bootcamp_grading_list.xml
   - customlist_sdc_curriculum_list.xml
   - customrecord_sdc_bootcamp_cur.xml
   - customrecord_sdc_bootcamp_empstatus.xml
   - customrecord_sdc_bootcamp.xml

9 Validate the project.

Import Instances

10 To import instances, right-click the customrecord_sdc_bootcamp_cur.xml file (under the Objects folder) and navigate to **NetSuite** > **Update Custom Record with Instances from Account**.

11 Click **OK** to continue.

12 Repeat step 11 for the customrecord_sdc_bootcamp_empstatus.xml file.

13 Open the manifest.xml file and delete all objects between <objects> and </objects>

14 Add the following between <objects> and </objects>

```
<object>customlist_sdc_bootcamp_grading_list.sdc_bg_a</object>
<object>customlist_sdc_bootcamp_grading_list.sdc_bg_exc</object>
<object>customlist_sdc_bootcamp_grading_list.sdc_bg_f</object>
<object>customlist_sdc_bootcamp_grading_list.sdc_bg_g</object>
<object>customlist_sdc_bootcamp_grading_list.sdc_bg_ni</object>
<object>customlist_sdc_bootcamp_grading_list.sdc_bg_vg</object>

<object>customlist_sdc_curriculum_list.sdc_cur_support</object>
<object>customlist_sdc_curriculum_list.sdc_cur_test</object>
<object>customlist_sdc_curriculum_list.sdc_cur_webdev</object>
<object>customlist_sdc_curriculum_list.sdc_cur_netadmin</object>
<object>customlist_sdc_curriculum_list.sdc_cur_infosec</object>
```

15 Validate the project.

Edit Instances

16 To edit an instance, open the customrecord_sdc_bootcamp_empstatus.xml file.

17 Scroll down and locate the start of the <instances> tag.

18 Locate **5% salary increase** and change it to **10% salary increase**.

19 Locate **25% salary increase** and change it to **20% salary increase**.

20 **Save** the file.

21 Validate and deploy the project.

Test

22 To test, log in to NetSuite and navigate to **Lists** > **Employees** > **Bootcamp File** > **New**.

23 Provide the needed information and select **Excellent** for the Final Grade.

24 Click **Employee Status** and ensure that 10% and 20% salary increases are shown.

EXERCISE 02: Inspect and Import Saved Search

Scenario: Saved searches cannot be edited in SDF, but you can back up saved searches locally. This exercise demonstrates how SDF handles saved searches.

Import a Saved Search

1. To import a saved search, in Eclipse, right-click **Objects** and navigate to **NetSuite** > **Import Custom Objects from Account**. The Import Wizard opens.
2. Review account details and click **Next**.
3. Click **Add/Remove** on Object Types.

   Note: Object Types allows for filtering results according to types. If you know the script ID for the object, use **Script ID Contains** to filter results.

4. From Available Object Types, highlight **savedsearch** and click **>**.
5. Click **Ok**.
6. Click **Search**.
7. Check the first saved search from the results list and click **Finish**.
8. Add the SERVERSIDESCRIPTING feature with required=”true” to manifest.xml.

   ```
   <feature required="true">SERVERSIDESCRIPTING</feature>
   ```

9. Open the customsearch1.xml file and take note of the definition.

   Note: In SDF, saved searches are system-generated. You cannot manually input criteria or results. Instead, edit the saved search using the **Edit Custom Object in Account** feature.

EXERCISE 03: Compare Local and Account Files

Scenario: In taking over a new SDF project, you notice that the project does not have proper documentation. Unsure if the local copy is up-to-date, you can proceed by importing the entire project again, but there could still be customizations considered for deployment. The best way to verify this is to compare the local and account files.

Change Values

1. To change values, in Eclipse, open the customlist_sdc_bootcamp_grading_list.xml file.
2. Locate and change **Excellent** to **Superior**.
3. Locate and change **Very Good** to **Excellent**.
4. **Save** the changes.

Compare XML

5. To compare .xml, right-click the customlist_sdc_bootcamp_grading_list.xml file.
6. Navigate to NetSuite and click **Compare Custom Object from Account Version…**. A split window opens both the local and account copies.
7. Compare the differences between both copies.
8. In the center of the window, a tiny square connects the two copies. Click this to manually update each line of the local copy with the account copy.

EXERCISE SOLUTIONS

EXERCISE 1: Import Bootcamp Bundle

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Bootcamp Bundle Exercise 01 Solution.zip**
2. Import to Eclipse.

EXERCISE 02: Inspect and Import Saved Search

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Bootcamp Bundle Exercise 02 Solution.zip**
2. Import to Eclipse.

EXERCISE 03: Compare Local and Account Files

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Bootcamp Bundle Exercise 03 Solution.zip**
2. Import to Eclipse.

CREATING CUSTOM OBJECTS IN ACCOUNT

Student Exercises	
01	Customize a Transaction Form
02	Create a Transaction Body Field
03	Delete a Custom Object

EXERCISE 01: Customize a Transaction Form

Scenario: Your company needs the Order Form modified to better meet its needs. The Order Form will become the preferred form used by sales representatives. Since manually creating a Transaction Form XML file in SDF is not possible, use the account to customize the transaction form.

Prerequisite: Create a new account customization project and name it “Order Project.”

Customize a Transaction Form

1. To customize a transaction form, expand the Order Project in Eclipse.
2. Right-click **Objects** and navigate to **NetSuite** > **New Custom Object in Account** > **Forms** > **Transaction Form**.
3. If prompted, select your account details and click **Ok**. The Custom Transaction Forms window opens.
4. Click **Customize** on Standard Order. The Custom Transaction Form window opens.
5. Input the values shown in the table below for the corresponding fields.

Field	Value
NAME	SDR Order Form for Sales
ID	_sdr_order_form
Tabs: Custom	Uncheck
Tabs: System Information	Uncheck
Tabs: Milestones	Uncheck
Roles: Sales Rep - Basic	Check
Roles: Sales Rep – US East	Check

6. Click **Save**.
7. Close the Custom Transaction Forms window.
8. Deselect all in the New Transaction Form, then check the custform_sdr_order_form file.
9. Click **OK**.

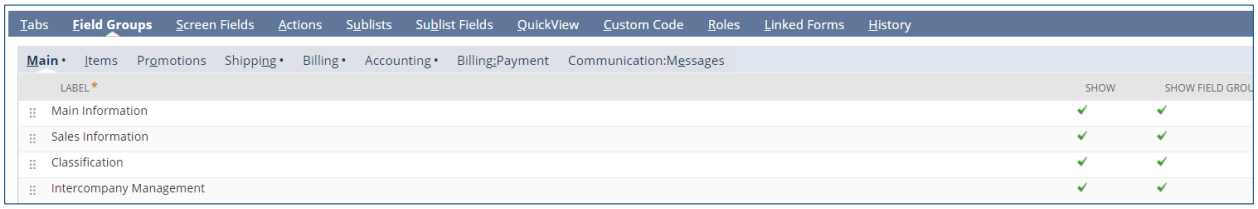
Note: This imports the custom form into the project, allowing you to edit the transaction form within SDF.

10. Right-click the manifest.xml file and select to **Add Dependency References to Manifest**.
11. Several warnings will appear under the **Problems** tab. These can be safely ignored.
12. Edit the custform_sdr_orderform.xml file.
13. Locate the <fieldGroup scriptid="primaryinformation"> tag.
14. Change the label from **Primary Information** to **Main Information**.
15. **Save** the file.
16. Validate and deploy the project.

Verify Changes

17. To verify changes, log into NetSuite and navigate to **Customization** > **Forms** > **Transaction Forms**.
18. In the Custom Transaction Form, click **Edit** on **SDR Order Form for Sales**.
19. Click the **Field Groups** subtab.

20 Verify that the Primary Information is successfully changed to Main Information.



***Field Groups** subtab*

EXERCISE 02: Create a Transaction Body Field

Scenario: Your company needs to add a checkbox to identify if an order is urgent. In this exercise, add a transaction body field.

Create an Urgent Field

1 To great a transaction body field, in Eclipse, right-click **Objects** and navigate to **NetSuite** > **New Custom Object in Account** > **Lists, Records, & Fields** > **Transaction Body Field**.

2 In the Transaction Body Field window, set the values for the transaction body field using the table below.

Property	Value
Label	Urgent
ID	_sdr_urgentfield
Type	Check Box
Applies To:	SALE
Display: Subtab	Main

3 Click **Save**.

4 Close the Transaction Body Field window.

5 Select custbody_sdr_urgentfield and click **OK**. The XML file generates.

6 In NetSuite, navigate to **Transactions** > **Sales** > **Enter Orders**.

7 Verify that the Urgent checkbox is available in Classification Field Group.

EXERCISE 03: Delete a Custom Object

Scenario: Your company decided to remove the Urgent field on sales orders. In this exercise, remove the custom object.

Delete the Urgent Field

1. To delete urgent field, in Eclipse, expand **Objects**.
2. Right-click custbody_sdr_urgentfield.xml and navigate to **NetSuite** > **Edit Custom Object in Account…**.
3. Click **Actions** > **Delete**.
4. A message may appear. Click **OK**.
5. Close the Transaction Body Field window.
6. A Problem Occurred message appears. Click **OK**.
7. Right-click the custbody_sdr_urgentfield.xml file and click **Delete**.

Note: This prevents deploying the field again.

EXERCISE SOLUTIONS

EXERCISE 01: Customize a Transaction Form

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Order Project Exercise 01 Solution.zip**
2. Import to Eclipse.

EXERCISE 02: Create a Transaction Body Field

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Order Project Exercise 02 Solution.zip**
2. Import to Eclipse.

EXERCISE 03: Delete a Custom Object

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Order Project Exercise 03 Solution.zip**
2. Import to Eclipse.

DEVELOPING SUITEAPPS

Student Exercises	
01	Create a New SuiteApp Project
02	Reference an Object from another SuiteApp

EXERCISE 1: Create a New SuiteApp Project

Scenario: In this exercise, develop a SuiteApp project from scratch.

Create an Account Customization Project

1. Navigate to **File** > **New** > **SuiteCloud Project**.
2. An option to select the SDF project type appears. Choose **Account Customization** and click **Next**.
3. Set the following properties:

   Project Name: SuiteApp Reference Demo

   SuiteScript Version: 2.0
4. Click **Finish**.

Import Record Types

5. Right-click **Objects** and navigate to **NetSuite** > **Import Custom Objects from Account…**.
6. Review account details and click **Next**.
7. Click **Add/Remove** on Object Types.
8. From Available Object Types, highlight **customrecordtype** and click **>**.
9. Click **Ok**.
10. Click **Search**.
11. The record type customrecord_sdr_foodrecipe_sa is not shown in the list since it is a custom record type in a SuiteApp.
12. Click **Cancel**.

Create a SuiteApp Project

13 Navigate to **File** > **New** > **SuiteCloud Project**.

14 An option to select the SDF project type appears. Choose **SuiteApp** and click **Next**.

15 Set the following properties:

Publisher ID: com.netsuite

Project ID: referencedemo

Project Name: SuiteApp Reference Demo

Project Version: 1.0.0

SuiteScript Version: 2.0

16 Click **Finish**.

Import Record Types

17 Right-click **Objects** and navigate to **NetSuite** > **Import Custom Objects from Account…**.

18 Review account details and click **Next**.

19 Click **Add/Remove** on Object Types.

20 From Available Object Types, highlight **customrecordtype** and click **>**.

21 Click **Ok**.

22 Click **Search**.

23 The record type customrecord_sdr_foodrecipe_sa is still not shown in the list.

24 Click **Cancel**.

EXERCISE 2: Reference an Object from another SuiteApp

Update manifest.xml and Deploy

1. Add to the dependencies section of manifest.xml the application from which the custom record has been deployed and the record type in the application.

```
<manifest projecttype="SUITEAPP">
    <publisherid>com.netsuite</publisherid>
    <projectid>referencedemo</projectid>
    <projectname>SuiteApp Reference Demo</projectname>
    <projectversion>1.0.0</projectversion>
    <frameworkversion>1.0</frameworkversion>
    <dependencies>
        <applications>
            <application id="com.netsuite.suiteappdemosetup">
                <objects>
                    <object>customrecord_sdr_foodrecipe_sa</object>
                </objects>
            </application>
        </applications>
    </dependencies>
</manifest>
```

2. Deploy the project.

EXERCISE SOLUTIONS

EXERCISE 1: Create a New SuiteApp Project

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **com.netsuite.referencedemo Exercise 01 Solution.zip**
2. Import to Eclipse.

EXERCISE 2: Reference an Object from another SuiteApp

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **com.netsuite.referencedemo Exercise 02 Solution.zip**
2. Import to Eclipse.

MANAGING SUITESCRIPT

Student Exercises	
01	Create a SuiteScript File
02	Define Script Record and Script Deployment
03	Deploy and Test

EXERCISE 01: Create a SuiteScript File

Scenario: A hands on demonstration of managing SuiteScript files with SDF.

1. **File** > **New** > **SuiteCloud Project**.
2. Choose **Account Customization** and click **Next**.
3. Set the following properties:
   - **Project Name**: Managing SuiteScript
   - **SuiteScript Version**: 2.0
4. Click **Finish**.
5. Expand the **File Cabinet** folder.
6. Right click the **SuiteScripts** folder and select **New** > **SuiteScript File**.
7. Set the properties according to the table below.

Property	Value
Script Type	User Event Script
Script Filename:	UserEventScript.js

8. Click **Finish**.
9. Open UserEventScript.js and locate the beforeLoad function:

```
function  beforeLoad(scriptContext) {

}
```

10 Replace the beforeLoad function with this:

```
function beforeLoad(scriptContext) {
   if (scriptContext.type === scriptContext.UserEventType.CREATE) {
       var customRecord = scriptContext.newRecord;
       customRecord.setValue({
          fieldId : 'custrecord_msg',
          value : 'My first account customization project'
       });
   }
}
```

11 Save the file.

EXERCISE 02: Define Script Record and Script Deployment

Scenario: A hands on demonstration of managing SuiteScript files with SDF.

1 Right-click the project and navigate to **New** > **Custom Object File**.

2 Set the properties according to the table below.

Property	Value
Type:	Script – User Event
Filename:	customscript_helloworld.xml

3 Click **Finish**.

4 In customscript_helloworld.xml, set the following:

```
<name>Hello World

<scriptfile>[/SuiteScripts/UserEventScript.js]

<scriptdeployment scriptid=”customdeploy_helloworld”>

<recordtype>[customRecord_helloworld]
```

Ensure you have end tags for each one.

5 Save the file.

6 Right-click the manifest.xml file and navigate to **NetSuite** > **Add Dependency References to Manifest**.

7 Validate the project.

EXERCISE 03: Deploy and Test

Scenario: A hands on demonstration of managing SuiteScript files with SDF.

1. Deploy the project.
2. In NetSuite, navigate to **Customization** > **Lists, Records, & Fields** > **Record Types**.
3. Select **New Record** for the **Hello World** record type.
4. The MESSAGE field should contain “My first account customization project”.

EXERCISE SOLUTIONS

EXERCISE 01: Create a SuiteScript File

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Managing SuiteScript Exercise 01 Solution.zip**
2. Import to Eclipse.

EXERCISE 02: Define Script Record and Script Deployment

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Managing SuiteScript Exercise 02 Solution.zip**
2. Import to Eclipse.

EXERCISE 03: Deploy and Test

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **Managing SuiteScript Exercise 03 Solution.zip**
2. Import to Eclipse.

PUBLISHING AND MANAGING SDF PROJECTS

Student Exercises	
01	Manage Deployment Files
02	Apply Content Protection
03	Bundle a SuiteApp Project

EXERCISE 01: Manage Deployment Files

Scenario: When implementing SDF for your company, selectively upload files for easier team management. In this scenario, objects assigned to Willy are still in development and are not to be deployed. Only deploy files created by Jed. As the release engineer in this exercise, deploy objects created by Jed.

Prerequisite: Download the SuiteDreams Center SuiteApp project (com.netsuite.suitedreamscenter) and import to your SuiteCloud IDE.

Organize SDF Folders

1. To organize SDF folders, open the imported project in Eclipse by navigating to **File** > **Import** > **Existing Projects into Workspace** > **Select Archive File** > **com.netsuite.suitedreamscente.zip**
2. Under com.netsuite.suitedreamscenter, right-click **Objects** and navigate to **New** > **Folder**.
3. Enter **Jed** as the folder name and click **Finish**.
4. Create another folder titled **Willy**.
5. Drag the following objects to the **Jed** folder.
   - custcentercategory_sdr_cat_enablement
   - custcentercategory_sdr_cat_forecasting
   - custcentercategory_sdr_cat_projmgmt
   - custcentertab_sdr_sd_tab
   - customrecord_sdr_sdc_enablement
   - customrecord_sdr_sdc_forecasting
   - customrecord_sdr_sdc_projmgmt

6. Drag the remaining objects into the **Willy** folder.
7. Expand **File Cabinet** > **SuiteApps** > **com.netsuite.suitedreamscenter**.
8. Create new folders, one titled **Jed** and another titled **Willy**.
9. Drag all files with a .js extension into the **Willy** folder.
10. Fix any errors by updating the paths of the script file to the Willy folder..

Update Deploy.xml

11. To update, open the deploy.xml file.
12. Update the path of files and objects to point to the **Jed** folder.

> **Note:** Specifying paths in the deploy.xml file allows you to selectively upload a specific folder. Organize SDF objects according to developers, project, functionality etc.

Deploy the Project

13. **Save** and deploy the project.
14. Review any **Deployment Logs** and note that only objects within the Jed folder are deployed.

EXERCISE 02: Apply Content Protection

Scenario: As the release engineer of SuiteDreams, set up content protection to either hide or lock customizations. In this exercise, lock custom records before bundling the SuiteApp project.

Create an Installation Preference

1. To apply content protection, right-click **com.netsuite.suitedreamscenter** and navigate to **New** > **Installation Preference File**.
2. Select **Locking Preference** for the Type.
3. The file name automatically populates.
4. Expand the Project and ensure that the InstallationPreference folder is selected.
5. Click **Finish**.

6 Edit the locking.xml file under the InstallationPreference folder.

7 Set the <defaultAction> tag to **UNLOCK**.

8 Enter **LOCK** for the <apply> tag.

9 Under the <applyaction> tag, create an <object> tag for the following objects:

- customrecord_sdr_sdc_enablement
- customrecord_sdr_sdc_forecasting
- customrecord_sdr_sdc_projmgmt

10 Save the locking.xml file and create new InstallationPreference file.

11 Select **Hiding Preference** and apply to the InstallationPreference folder.

12 Click **Finish**.

13 Edit hiding.xml.

14 Set the <defaultAction> tag to **UNHIDE**.

15 Enter **HIDE** for the <applyaction> tag.

16 **Save** the file.

To hide additional files and folders, add <path> where necessary. This action is skipped for the purposes of this exercise.

17 Deploy the project to the account and check the Apply Content Protection checkbox.

Note: Both hide.xml and lock.xml must be present when creating content protection. If either is missing, you will receive an error during validation and deployment.

18 Log in to NetSuite and navigate to **Lists, Records & Fields** > **Record Types**.

19 Check if there is a lock icon [🔒] on the **Enablement**, **Forecasting** and **Project Management** tabs. If so, content protection is applied.

20 If there is no locking mechanism [🔒] on the above tabs, return to the locking.xml file and repeat the steps above.

EXERCISE 03: Bundle a SuiteApp Project

Scenario: After applying content protection, bundle the SuiteApp project in order to deploy it to accounts for which you are not the administrator.

Enable SuiteBundler

1. To enable SuiteBundler, navigate to **Setup** > **Company** > **Enable Features** > **SuiteCloud**.
2. Check CREATE BUNDLES WITH SUITEBUNDLER, under the SuiteBundler section.
3. **Agree** to the Terms of Service and **Save**.

Bundle the Project

4. To bundle the project, navigate to **Customization** > **SuiteBundler** > **Create Bundle**.
5. Enter **SDF Course: SuiteDreams Center_<yourname>** in the NAME field.
6. Click the **APP ID** dropdown and select your project.
7. Click **Next**. The Bundle Properties screen opens.
8. Add a description and upload documentation on the Bundle Properties screen but skip this step for this exercise and click **Next**.

   Note: With the project selected on the **APP ID** dropdown, there is no need to select Bundle contents manually.

9. Click **Next**.

You cannot change the locking preference on objects with content protection applied during SDF deployment.

10. Click **Save**.
11. Navigate to **Customization** > **SuiteBundler** > **Create Bundle** > **List**.
12. Under **Action**, hover to the Name icon [] and select **Set Availability**.
13. Change the LEVEL to either **Public** or **Shared**.
    - **Private** is only accessible to other accounts to which you have administrator access.
    - **Shared** allows account administrators you list in the Shared Account IDs to view and install a shared bundle.

- **Public** allows any NetSuite administrator to view and install the bundle from a deployment account and from the repository (if the bundle is copied there).

14 Click **Finish**.

EXERCISE SOLUTIONS

EXERCISE 01: Manage Deployment Files

Download from File Cabinet:

7. **SDF Files** > **Exercise Solutions** > **com.netsuite.suitedreamscenter Exercise 01 Solution.zip**
8. Import to Eclipse.

EXERCISE 02: Apply Content Protection

Download from File Cabinet:

1. **SDF Files** > **Exercise Solutions** > **com.netsuite.suitedreamscenter Exercise 02 Solution.zip**
2. Import to Eclipse.

USING THE COMMAND LINE INTERFACE

Student Exercises	
01	Set Up SDF CLI
02	Create an Account Customization Project
03	Upload Custom Object

EXERCISE 01: Set Up SDF CLI

Scenario: Start using your favorite code/text editor with SDF Command Line Interface (CLI) and avoid using Eclipse and WebStorm. In this exercise, configure an environment by properly installing SDF CLI on your desktop.

Prerequisites:

- Download SDF CLI (for Windows or for Mac) from Help Center

  SuiteCloud Platform > **SuiteCloud SDK** > **SuiteCloud CLI** > **SuiteCloud CLI for Java** > **CLI for Java Guide** > **Installing CLI for Java**
- Extract SDF CLI
- Download and install text editor (optional)

Set Up SDF CLI

1. To set up SDF CLI, create a folder on the desktop and name it as **SDFCLI**.
2. Extract the sdf-cli.zip file into the SDFCLI folder.
3. Open **Command Prompt/Terminal**.
4. Navigate to the SDFCLI folder using CMD commands.
5. At the SDFCLI directory, enter **sdfcli**. This verifies that SDF CLI is working properly working. If correctly configured, a list of commands appears on the terminal.
6. If the list of commands does not appear, ensure that you opened the correct directory.

Organize Workspace

7. Organize the workspace by creating a new folder in the SDFCLI folder and name it as **sdf-workspace**.

EXERCISE 02: Create an Account Customization Project

Scenario: With SDF CLI configured, start creating a project in the terminal. In this exercise, create an AC project and deploy it to your NetSuite Account.

Create a Project

1. To create a project, open the terminal and enter the following command:

   ```
   sdfcli createproject
   ```

2. A message appears requesting that you specify the -pd. If unsure of the commands, SDF CLI can identify the switches needed to successfully execute the command. Update the command above and enter the following:

   ```
   sdfcli createproject -pd <directory of workspace> -t ACCOUNTCUSTOMIZATION -pn HelloWorld
   ```

3. If successful, the following message appears.:

   ```
   Project <directory>\HelloWorld has been created.
   ```

Select a Default Project

4. To select the new project as the default, open the terminal, and enter the following command:

   ```
   sdfcli project -p <directory of workspace>\HelloWorld
   ```

Setting a default project helps you avoid entering the project directory at every deployment.

Validate the Project

5. Enter the following command in the terminal to validate the project against the account:

   ```
   sdfcli validate -account <TSTDRVxxx> -email <login email>
   ```

   ```
    -role 3 -url system.netsuite.com
   ```

6. When prompted, enter your password. If successful, you will receive no errors.

EXERCISE 03: Upload Custom Object

Scenario: In this exercise, create a custom object using the text editor and upload files with CLI.

Upload a Custom Object (in Text Editor)

1. Navigate to **SDF-workspace** > **HelloWorld** folder.
2. Create a text document.
3. Enter **custcentertab_helloworld.xml** as the file name.

The file extension must be .xml.

4. Edit the custcentertab_helloworld.xml file and enter the following code:

```
<centertab scriptid="custcentertab_helloworld">
    <allroles>T</allroles>
    <center>ALL</center>
    <label>HelloWorld</label>
</centertab>
```

5. **Save** the file.

Validate and Deploy

6. Validate the project in CLI by entering the following command:

```
sdfcli validate -account <TSTDRVxxx> -email <login email>
  -role 3 -url system.netsuite.com
```

7. When prompted, enter your password.
8. A Feature Dependency error shows. Fix the error by adding the dependency to the project manifest. Enter the following command:

```
sdfcli adddependecies -all
```

9. Type **YES** when prompted.
10. Deploy the project by entering the following command:

```
sdfcli deploy -account <TSTDRVxxx> -email <login email> -role 3 -url
system.netsuite.com
```

11. When prompted, enter your password.

12 Type **Y** to proceed.

13 Type **YES** to deploy the project to the account.

14 Log in to NetSuite and verify if the **Hello World** tab appears in the NetSuite Center.

HelloWorld tab in the NetSuite Center

APPENDIX A | GENERIC IDS

ID: -112 List/Record: Account

ID: -215 List/Record: Account Type

ID: -414 List/Record: Accounting Context

ID: -105 List/Record: Accounting Period

ID: -289 List/Record: Address

ID: -137 List/Record: Address Book

ID: -387 List/Record: Advanced PDF/HTML Template

ID: -138 List/Record: Allocation Schedule

ID: -278 List/Record: Allocation Type

ID: -232 List/Record: Amortization Type

ID: 10 List/Record: Annual Revenue

ID: -243 List/Record: Approval Status

ID: -408 List/Record: Bank Import History

ID: 72 List/Record: Bed Sizes

ID: -139 List/Record: Billing Class

ID: -140 List/Record: Billing Milestone

ID: -141 List/Record: Billing Schedule

ID: -234 List/Record: Billing Schedule Type

ID: 49 List/Record: Board Plan: Revenue Target

ID: -420 List/Record: Budget

ID: -136 List/Record: Buying Reason

ID: -135 List/Record: Buying Time Frame

ID: 12 List/Record: Buying Time Frame

ID: -22 List/Record: Call

ID: -24 List/Record: Campaign

ID: -142 List/Record: Campaign Audience

ID: -143 List/Record: Campaign Category

ID: -144 List/Record: Campaign Channel

ID: -229 List/Record: Campaign Email

ID: -226 List/Record: Campaign Event Response

ID: -227 List/Record: Campaign Event Type

ID: -145 List/Record: Campaign Family

ID: -146 List/Record: Campaign Offer

ID: -131 List/Record: Campaign Response Type

ID: -148 List/Record: Campaign Search Engine

ID: -149 List/Record: Campaign Subscription

ID: -150 List/Record: Campaign Vertical

ID: -23 List/Record: Case

ID: -151 List/Record: Case Issue

ID: -152 List/Record: Case Origin

ID: -153 List/Record: Case Priority

ID: -320 List/Record: Case Profile

ID: -132 List/Record: Case Status

ID: -154 List/Record: Case Type

ID: 79 List/Record: Category

ID: -228 List/Record: Category 1099 Misc

ID: -319 List/Record: Charge Billing Mode

ID: -101 List/Record: Class

ID: -212 List/Record: Commission Payment Type

ID: -271 List/Record: Commission Plan

ID: -270 List/Record: Commission Schedule

ID: 9 List/Record: Company Size

ID: 104 List/Record: Compatible Browsers

ID: -108 List/Record: Competitor

ID: -240 List/Record: Consolidated Rate Type

ID: -322 List/Record: ConsolidatedExchangeRate

ID: -6 List/Record: Contact

ID: -158 List/Record: Contact Category

ID: -157 List/Record: Contact Role

ID: -184 List/Record: Costing Method Type

ID: -159 List/Record: Country

ID: -160 List/Record: Credit Hold Override Type

ID: -122 List/Record: Currency

ID: -282 List/Record: Currency Symbol Placement

ID: -2 List/Record: Customer

ID: -109 List/Record: Customer Category

ID: -161 List/Record: Customer Message

ID: -102 List/Record: Department

ID: -375 List/Record: Device Id

ID: -241 List/Record: Edition

ID: -165 List/Record: Email Preference Type

ID: -120 List/Record: Email Template

ID: -4 List/Record: Employee

ID: 106 List/Record: Employee OS-Browser

ID: -166 List/Record: Employee Status

ID: -111 List/Record: Employee Type

ID: -9 List/Record: Entity

ID: -8 List/Record: Entity Group

ID: -223 List/Record: Entity Stage

ID: -104 List/Record: Entity Status

ID: -214 List/Record: Entity Type

ID: -167 List/Record: Entry Form

ID: -168 List/Record: Ethnicity

ID: -20 List/Record: Event

ID: -222 List/Record: Event Priority

ID: -169 List/Record: Event Reminder Duration

ID: -170 List/Record: Event Reminder Type

ID: -230 List/Record: Event Response Type

ID: -218 List/Record: Event Status

ID: -126 List/Record: Expense Category

ID: -124 List/Record: Field

ID: -213 List/Record: Field Type

ID: -147 List/Record: File Type

ID: 88 List/Record: Furniture

ID: -540 List/Record: General Token

ID: -310 List/Record: Generic Resource

ID: -324 List/Record: Incoterm

ID: 8 List/Record: Industry

ID: -266 List/Record: Inventory Number

ID: -10 List/Record: Item

ID: -506 List/Record: Item Location Configuration

ID: -106 List/Record: Item Type

ID: -224 List/Record: Language

ID: -237 List/Record: Locale

ID: -103 List/Record: Location

ID: -371 List/Record: Location Type

ID: -178 List/Record: Marital Status

ID: -865 List/Record: Memorized Transaction

ID: -179 List/Record: Merchant Account

ID: -281 List/Record: Negative Number Format

ID: -532 List/Record: News Item

ID: -400 List/Record: Nexus

ID: -180 List/Record: Note Type

ID: -280 List/Record: Number Format

ID: 103 List/Record: Operating Systems

ID: -31 List/Record: Opportunity

ID: -244 List/Record: Order Status

ID: -181 List/Record: Other Name Category

ID: -210 List/Record: Out of Stock Behavior Type

ID: -5 List/Record: Partner

ID: -182 List/Record: Partner Category

ID: -538 List/Record: Payment Card

ID: -539 List/Record: Payment Card Token

ID: -537 List/Record: Payment Instrument

ID: -183 List/Record: Payment Method

ID: 87 List/Record: Performance Review

ID: -261 List/Record: Period Subsidiary

ID: -220 List/Record: Phone Call Status

ID: -186 List/Record: Price Level

ID: -187 List/Record: Pricing Group

ID: -225 List/Record: Product Tax

ID: -7 List/Record: Project

ID: -211 List/Record: Project Constraint Type

ID: -287 List/Record: Project Expense Type

ID: -517 List/Record: Project Resource Role

ID: -27 List/Record: Project Task

ID: -321 List/Record: Project Template

ID: -528 List/Record: Project Time Approval Type

ID: -177 List/Record: Project Type

ID: -121 List/Record: Promotion Code

ID: -233 List/Record: Promotion Code Discount Type

ID: -205 List/Record: Quantity Pricing Schedule

ID: -206 List/Record: Quantity Pricing Type

ID: 102 List/Record: Quote Approval Statuses

ID: -123 List/Record: Record Type

ID: -188 List/Record: Resident Status

ID: -28 List/Record: Resource Allocation

ID: -301 List/Record: Resource Allocation Approval Status

ID: -216 List/Record: Revenue Commitment Status

ID: -189 List/Record: Revenue Recognition Schedule

ID: -231 List/Record: Revenue Recurrence Type

ID: -217 List/Record: Revenue Status

ID: 86 List/Record: Review Type

ID: -118 List/Record: Role

ID: -134 List/Record: Sales Readiness

ID: -162 List/Record: Sales Territory

ID: -119 List/Record: Saved Search

ID: -417 List/Record: Script

ID: -418 List/Record: Script Deployment

ID: -125 List/Record: Scripted Record Type

ID: -192 List/Record: Shipping Method

ID: -193 List/Record: Shipping Package

ID: -245 List/Record: Shipping Status

ID: -185 List/Record: Site Category

ID: -194 List/Record: Site Template

ID: -195 List/Record: State

ID: 46 List/Record: Subscriber Log

ID: -197 List/Record: Subscription Status

ID: -117 List/Record: Subsidiary

ID: -163 List/Record: Support Territory

ID: -21 List/Record: Task

ID: -264 List/Record: Task Item Status

ID: -219 List/Record: Task Status

ID: -128 List/Record: Tax Code

ID: -127 List/Record: Tax Period

ID: -239 List/Record: Tax Rounding Type

ID: -198 List/Record: Taxation Type

ID: -199 List/Record: Term

ID: -256 List/Record: Time

ID: 1 List/Record: Time Zones

ID: -238 List/Record: Timezone

ID: -30 List/Record: Transaction

ID: -171 List/Record: Transaction Form

ID: -164 List/Record: Transaction Status

ID: -100 List/Record: Transaction Type

ID: -221 List/Record: Unit

ID: -201 List/Record: Units Type

ID: -3 List/Record: Vendor

ID: -110 List/Record: Vendor Category

ID: -549 List/Record: Vendor-Subsidiary Relationship

ID: -202 List/Record: Visa Type

ID: -268 List/Record: Website

ID: 89 List/Record: White Shag Rug Sizes

ID: -203 List/Record: Win/Loss Reason

ID: -156 List/Record: Work Calendar

ID: -129 List/Record: Workflow

ID: -236 List/Record: Workflow Release Status

ID: -235 List/Record: Workflow Trigger Type

ID: -196 List/Record: Workplace