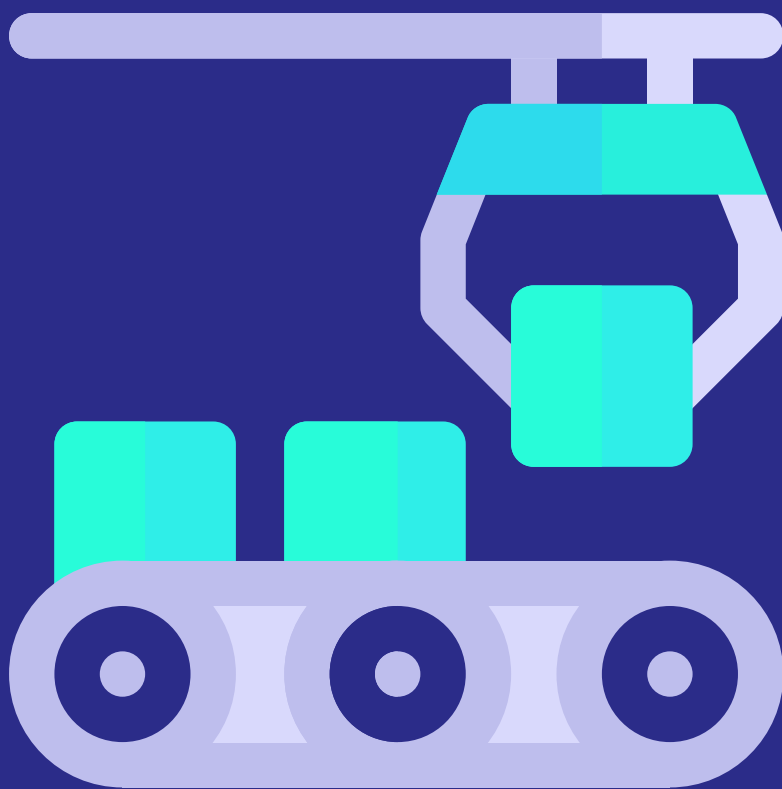


SuiteCloud Development Framework



eBook

Version 1.0



Techfino

AUTHORS

Bryan Willman

Luke Pirtle

RenHong Zhu

Nick Klug

PUBLISHED DATE

8/12/2019

LAST UPDATED

8/12/2019

VERSION HISTORY

Version: V1

Date: 8/12/2019

Description: Initial publication

Table of Contents

Techfino's SuiteCloud Development Framework	1
Audience and Purpose	4
Introduction	4
Guide to Account Customization with SDF	6
Understanding Customization XMLs	7
Customization Object XML	7
Manifest XML	8
SDF Development Environment Setup	8
IDE Integrations	8
Install Eclipse	9
Install SDF Plugin	10
Enable SDF and Create Custom Role for SDF in NetSuite	13
Configure Accounts for SDF Project in Eclipse	17
Change Project Settings	17
Add Customizations to SDF	20
Import by ScriptID	20
Import by Bundle ID	22
Manually Create Customizations	24
Add Dependency to SDF	25
Validation and Deploying SDF Project	26
Validate Project Against Account	26
Deploy SDF Project to Account	27
Sample: Deploy Mass Delete Script from One Account to Another	28
Background	28
Project and Git setup in Eclipse	31
Import Customizations and Validate before Deployment	41
Deploy to Target Account	48
Additional Notes for This Example	50
SDF Gotchas & Our Experience	51
Closing Remarks	53

Audience and Purpose

This guide is written for NetSuite developers and technical resources who are looking for documentation and step-by-step process guides for the SuiteCloud Developer Framework (SDF). After reading this ebook a user should have a good understanding of the framework, its advantages and shortcomings as a DEVOPS tool. The reader is guided through the process of setting up and configuring an SDF project for ongoing development and support of NetSuite customizations. Techfino has also included a sample SDF project to walk the user through a real-world deployment for demonstration purposes.

Introduction

Over the past few years NetSuite has been hard at work adding exciting features to the NetSuite development process with the introduction of the SuiteCloud Developer Framework or “SDF” as it’s most commonly abbreviated. SDF is in essence a platform-agnostic big brother to the Eclipse SuiteCloud IDE many developers are already quite familiar with. However, SDF is a much more mature and established tool that can deploy an entire NetSuite customization project or suiteapp instead of just the source code. Adoption of SDF has been quite slow to the NetSuite developer community due to its added complexity and lack of guides and documentation. This document is meant as a guide to help a developer understand what SDF is, its advantages and how to start using SDF through the examples below.

The precursor to SDF, SuiteCloud IDE, allows a developer to directly upload their SuiteScript source code files into the NetSuite File Cabinet. This streamlines ongoing development by allowing a developer to quickly push code changes to NetSuite without context / window switching to another app. However, it is far from ideal as a deployment tool for NetSuite developers and pales in comparison to other platforms’ deployment tools. NetSuite projects are more than just suitescript source code and require customization and configuration in the account to be setup and run. SuiteScripts are reliant on custom records, custom fields, saved searches, workflows etc. which are not supported by SuiteCloud IDE and require manual creation or another tool working in tandem.

Most NetSuite partners utilize SuiteBundler to fill this deployment gap. While it may bridge this gap, in practice SuiteBundler can be incredibly obtuse and difficult to use at times. It often leaves a heavy reliance on documentation and manual process. SuiteBundler doesn’t have version control support and components aren’t linked to specific file versions which can lead to conflicting code and deployment issues.

SDF aims to address the gaps and pain points of the alternatives by giving a holistic deployment tool that can be used to create and deploy projects seamlessly. To start off, SDF is available for a few supported IDE's (Eclipse, Webstorm) just like its predecessor but can also be used as a command line interface (CLI). This addition allows SDF to be adapted for community plugins and deployment pipelines such as Jenkins. This allows it to be used for small agile projects or full blown application development. SDF also supports account configuration and customization by serializing definitions into xml. In practice, this allows entire projects to be updated and installed to new accounts by a single user without having to leave their IDE. SDF centralizes the code, customization and configuration which eliminates the tedious documentation necessary to keep everything in sync. This also allows for entire projects to be version controlled as opposed to only its source code.

	SuiteCloud IDE	SuiteBundle	SDF
Supports SuiteScript	✓	✓	✓
Supports Configuration		✓	✓
Centralized Project		✓	✓
Can be deployed from an IDE / External Tool	✓		✓
Can be paired with version control	✓		✓
Typical time spent validating deployment	Instant (code out of sync is marked in IDE)	30 min - 5 hours	< 5 min
Typical time spent deploying	< 5 min	< 20 min	< 5 min

NetSuite Customization Deployment Tool Comparison

The biggest drawback to SDF is that it is a big upgrade. NetSuite has added a lot recently to a mostly barren deployment landscape. It's early stages have been buggy with lacking support and documentation. However, since SDF's release, a large number of these issues have been addressed. At Techfino, we are now using SDF for all new development projects and encourage other NetSuite Partners and Customers to do the same.

Guide to Account Customization with SDF

As presented in the Techfino [Guide to SuiteBundling in NetSuite](#) by Nick Klug, we assist customers with NetSuite deployments involving the transfer of both configuration and customizations created in one NetSuite account to another. We have been shifting into utilizing the SuiteCloud Development Framework (SDF for short) to help with this process for a variety of reasons which are outlined below. In the following section, we will investigate SDF for managing customizations in multiple environments and deployment. We will also be discussing how SDF differs from SuiteBundling and some of the things we have learned over the past year.

For those not familiar with SDF, it can best be thought of as a new framework in which both configurations and customization objects are represented in XML format. Some of the benefits of this new process include:

- **Greater Control** - the files have a much more precise state with SDF. The XML that is present in the SDF project is exactly what is imported into the target system.
- **Version Control** - the XML files can also be stored in a version control system, such as Github or bitbucket. This allows for analysis of changes in the life cycle of a project and can allow for reverting to a previous customization.
- **Build-Automation** - while this is not fully baked as of now, Techfino has discovered a technology stack leveraging SDF that allowed us to successfully automate the deployment of versioned changes from one account to another. SDF is an exciting, enabling technology that will allow NetSuite to advance into more of a mature development and automated build deployment platform allowing for Continuous Integration and Test Automation to follow.

When starting to utilize the SDF platform, an easy way to start is to simply create a few basic scripts and fields within NetSuite. Then, import them into SDF and examine the new entries in the SDF project. This gives a good starting point for your customization project. See the Sample Mass Delete Script Deployment section for a "Hello World" example of deploying a simple development package via SDF.

Understanding Customization XMLs

Customization Object XML

SDF works by representing a customization record in NetSuite with an XML file. This works both ways - extracting an existing customization from an account and creating an XML, and pushing the XML into a target account, creating the customization there. The below example shows the XML object for a User Event script. It contains references to all the fields on that customization record within NetSuite. This includes the scriptID, description, what functions it executes on, and whether or not it is active.

Node	Content
▼ [e] useeventscript	
ⓐ scriptid	customscript1
[e] aftersubmitfunction	saveReview
[e] beforeloadfunction	
[e] beforesubmitfunction	
[e] description	
[e] isinactive	F
[e] name	Link Product Review to Inventory Item
[e] notifyadmins	F
[e] notifyemails	
[e] notifyowner	T
[e] notifyuser	F
[e] scriptfile	[/SuiteScripts/linkReviewToItem_v3.js]
▼ [e] scriptdeployments	
> [e] scriptdeployment	

Additionally, it also details script deployments at the parameters it uses and what their values are set to.

▼ [e] scriptdeployments	
▼ [e] scriptdeployment	
ⓐ scriptid	customdeploy1
[e] allemployees	F
[e] allpartners	F
[e] allroles	T
[e] audslectrole	
[e] eventtype	
[e] executeinwebstore	T
[e] isdeployed	T
[e] loglevel	DEBUG
[e] recordtype	[scriptid=customrecord_product_review]
[e] runasrole	ADMINISTRATOR
[e] status	RELEASED

Manifest XML

```
1 <manifest projecttype="ACCOUNTCUSTOMIZATION">
2   <projectname>Test Project</projectname>
3   <frameworkversion>1.0</frameworkversion>
4   <dependencies>
5     <features>
6       <feature required="true">SERVERSIDESCRIPING</feature>
7       <feature required="false">CREATESUITEBUNDLES</feature>
8       <feature required="false">CRM</feature>
9     </features>
10   <files>
11     <file>/SuiteScripts/script_core_lib.js</file>
12   </files>
13 </dependencies>
14 </manifest>
```

The manifest for the project contains the requirements for the manifest to be deployed in an environment. For example, if you want to use SDF to deploy SuiteScripts to the target account, then the target account is required to have Server Side Scripting enabled.

One of the greatest strengths of SDF compared to SuiteBundling is allowing for the deployment of objects with dependencies without having to include those dependencies. This is assuming they may have already been deployed via another SuiteBundle. In practice, workflows are some of the most difficult objects to move with a SuiteBundle. This is because workflows often reference multiple customizations and with a SuiteBundle, each related customization will be included automatically. With each migration to an environment, there is a possibility of unintentionally changing the customizations in the target environment with the bundled customizations.

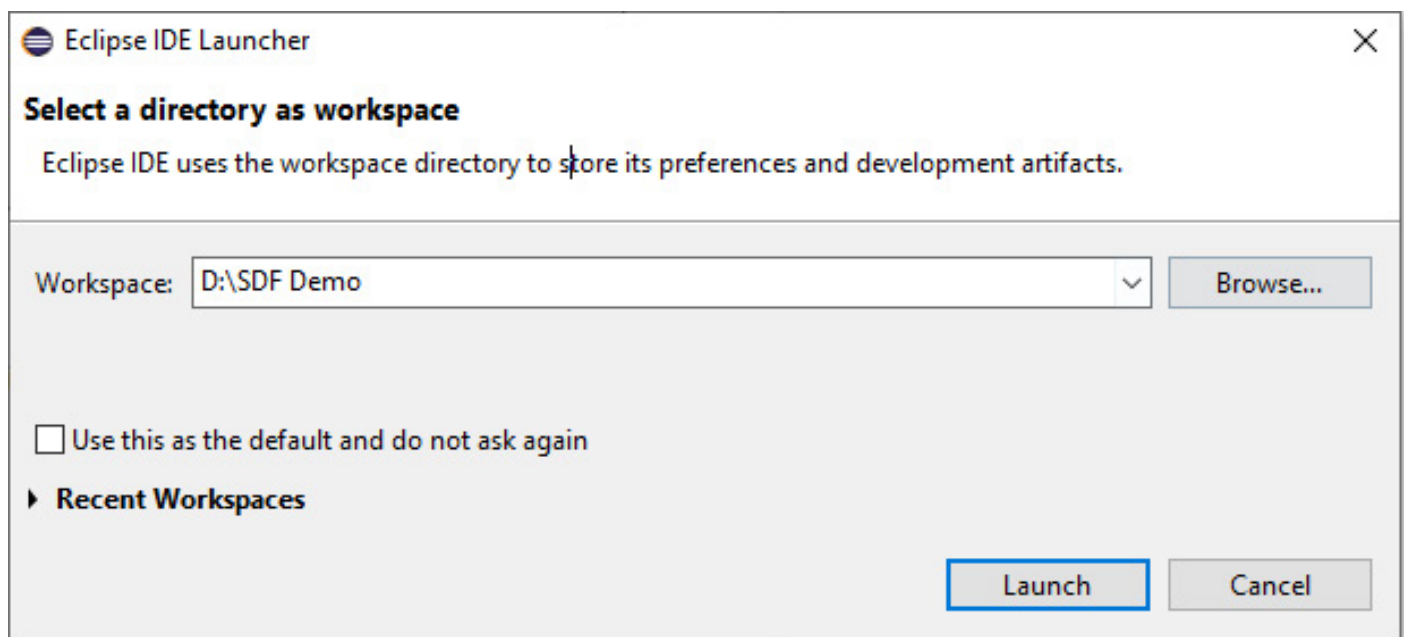
SDF Development Environment Setup

IDE Integrations

SDF officially integrates with Eclipse and WebStorm, with Eclipse being the more mature of the two options. In the following sections, we will be using Eclipse for demonstration purposes. Below you will find the detailed instructions to setup Eclipse with SDF. You can also follow NetSuite provided instructions (See SuiteAnswers ID #10315 for Setting Up SuiteCloud IDE Plug-in for Eclipse).

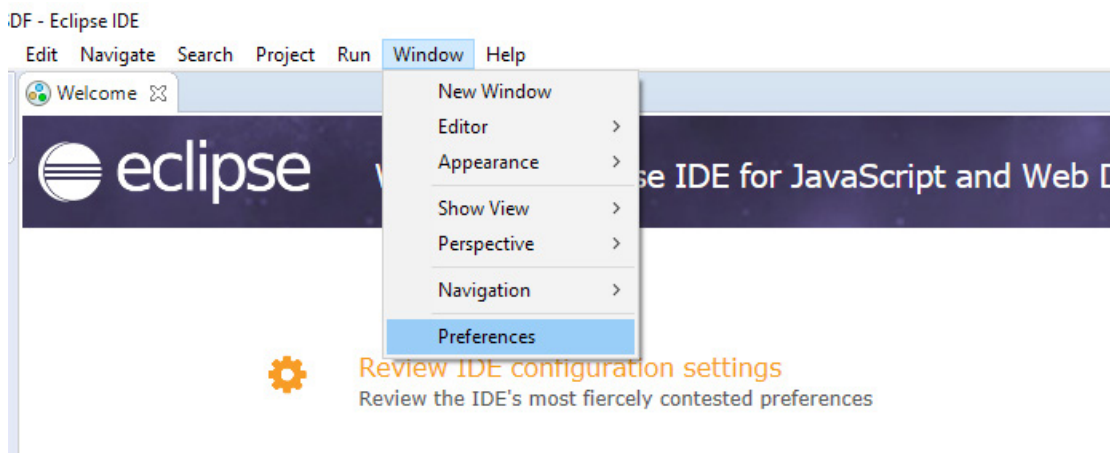
Install Eclipse

To get started with SDF, you will have to download Eclipse first. NetSuite recommends using Eclipse Mars. (02/26/2016 package. Link: <https://www.eclipse.org/downloads/packages/release/mars/2/eclipse-ide-java-ee-developers>.) Although, we haven't experienced any issues with the latest version of Eclipse IDE. J2EE or Javascript/Web Developer version both works. Link: <https://www.eclipse.org/downloads/packages/>. Once installed, you can change the workspace to a different path. This will make access to the folder easier.

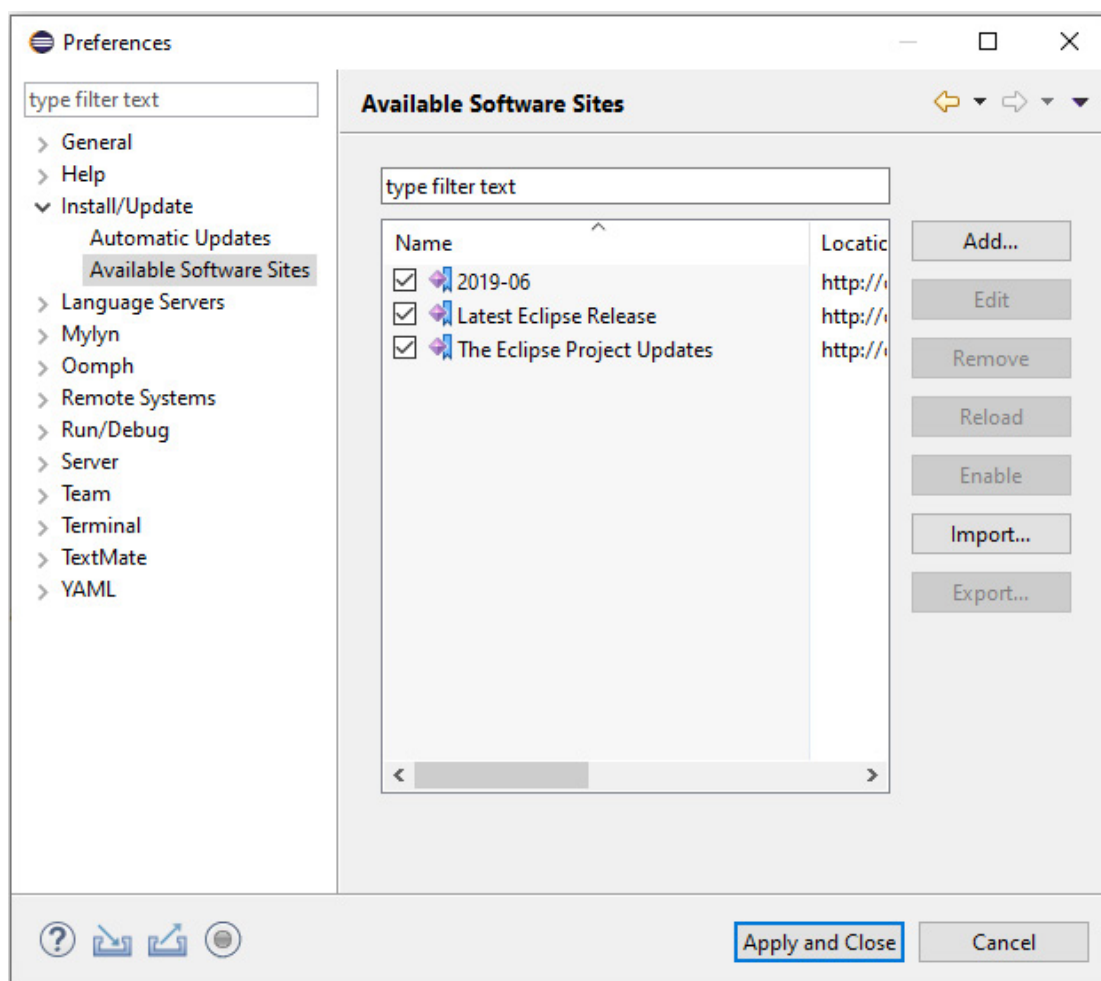


Install SDF Plugin

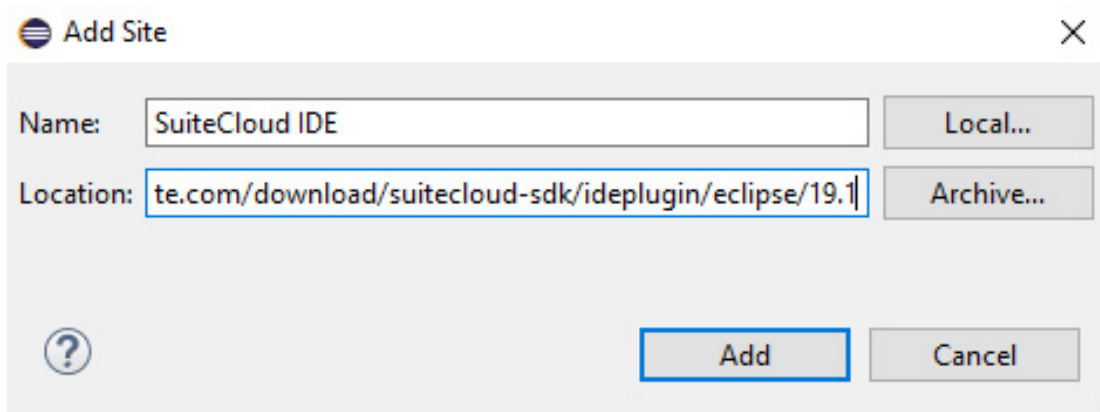
1. Run Eclipse, Under Window tab, find Preferences



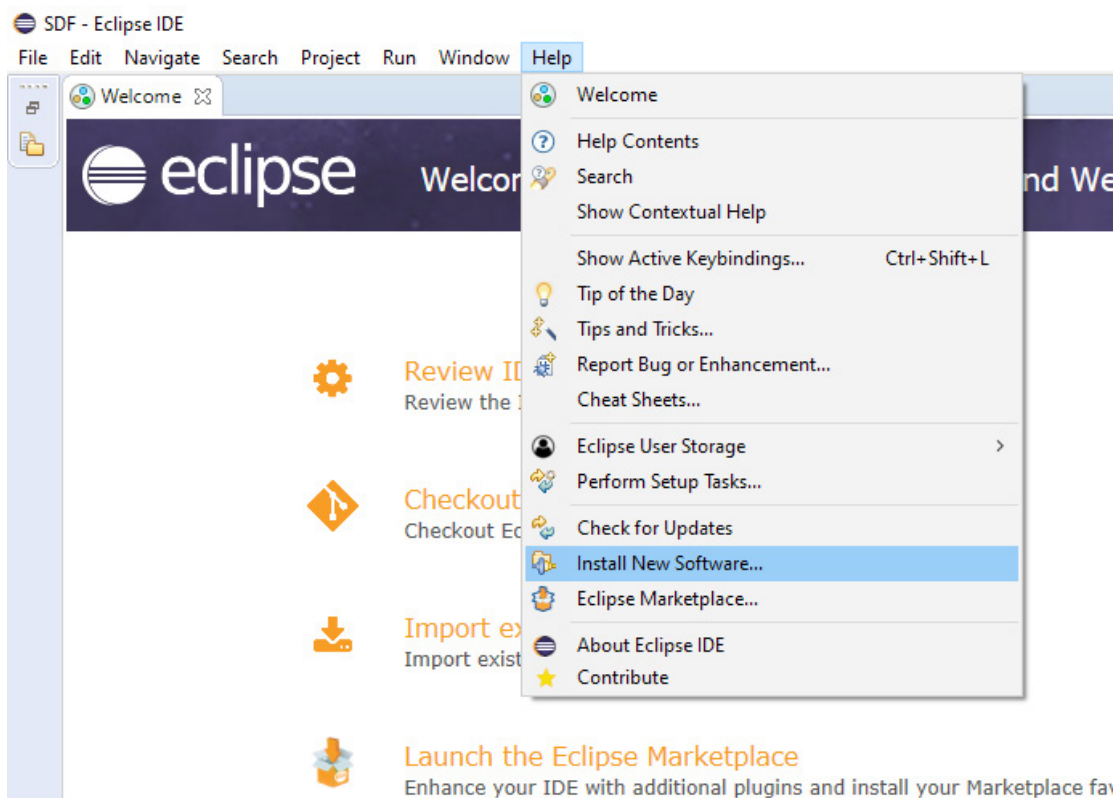
2. In the opened window go to Install/Update > Available Software Site



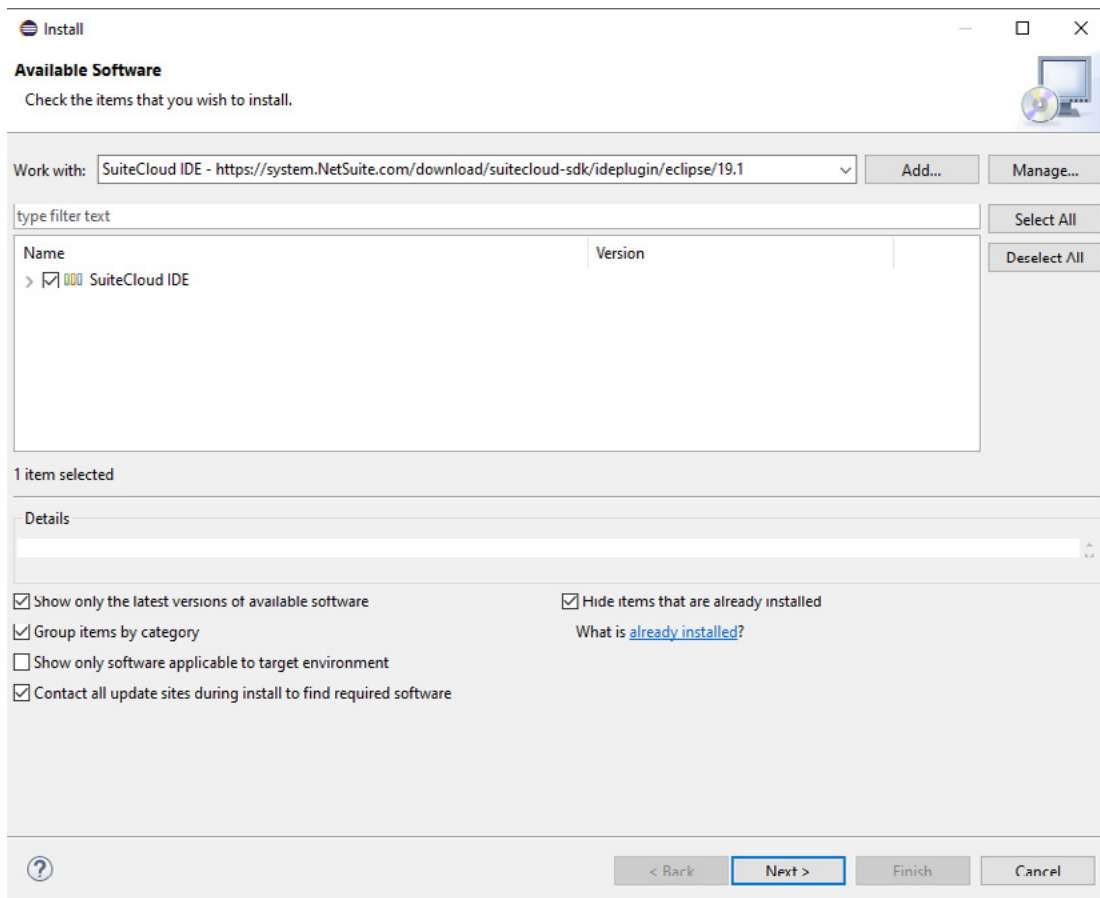
3. Click Add and enter the following information
 - Name: SuiteCloud IDE
 - Location: <https://system.NetSuite.com/download/suitecloud-sdk/ideplugin/eclipse/19.1> (This address changes with releases, so if you are using a builder later than 2019.1 you might need to lookup the address)



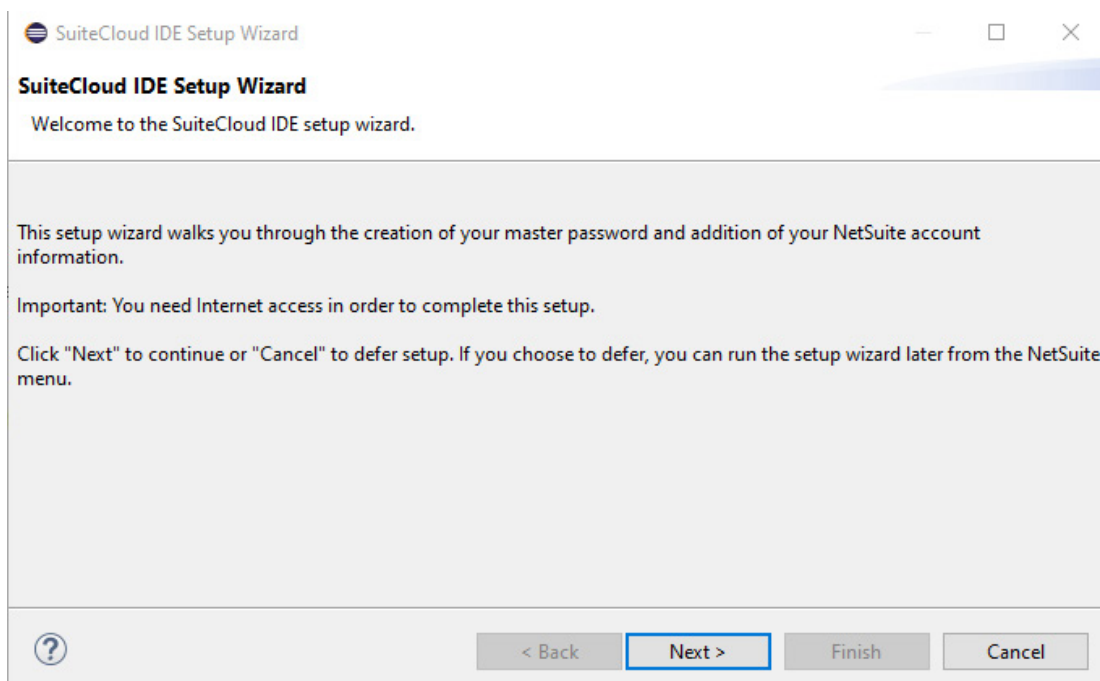
4. Apply and Close
5. Go to Help tab, click Install New Software



6. In the opened window, select SuiteCloud under Work with field and verify the address. Check the SuiteCloud IDE checkbox in the list and follow the Eclipse wizard to complete installation.



7. When prompted, restart Eclipse
8. If installation is successful, user will see the following dialog after restart

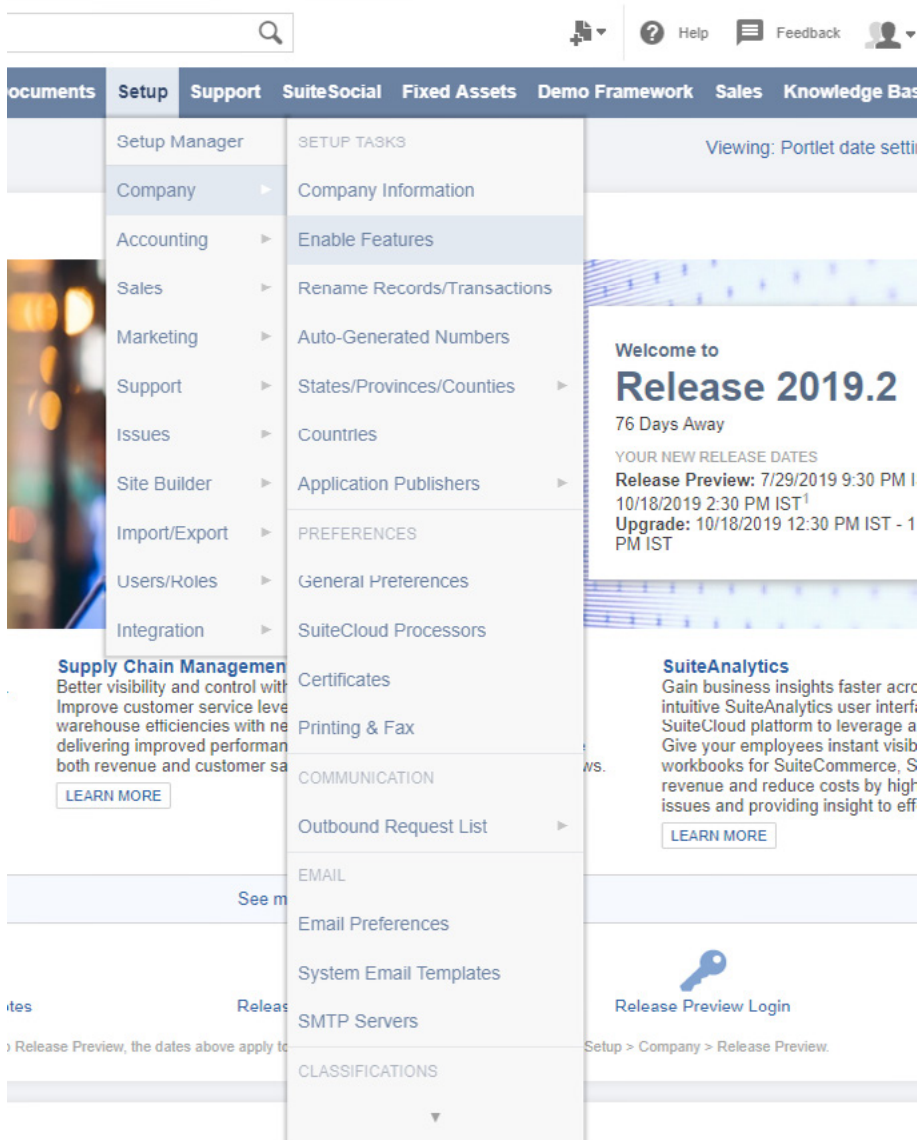


9. Setup master password for SuiteCloud IDE. Once the password is set up, user will have to use the master password to enable SuiteCloud plugin functions for all SuiteCloud projects

Enable SDF and Create Custom Role for SDF in NetSuite

Because NetSuite doesn't come with the SDF feature enabled by default, an Administrator will have to manually enable the feature. A new custom role should be created for SDF since the Administrator role and Developer role will encounter an issue with 2FA when using SDF. The new SDF role should be created from the Developer role for SDF purposes.

1. Login to NetSuite using Administrator role and go to Setup > Company > Enable Features



2. In the new page, click on SuiteCloud tab

Enable Features

Save

Cancel

Reset

Company Accounting Tax Transactions Items & Inventory Employees CRM Analytics Web Presence SuiteCloud

VIEW SUITECLOUD [TERMS OF SERVICE](#)

3. Scroll all the way to the bottom and check SDF option, agree to the terms and save the change

SuiteCloud Development Framework

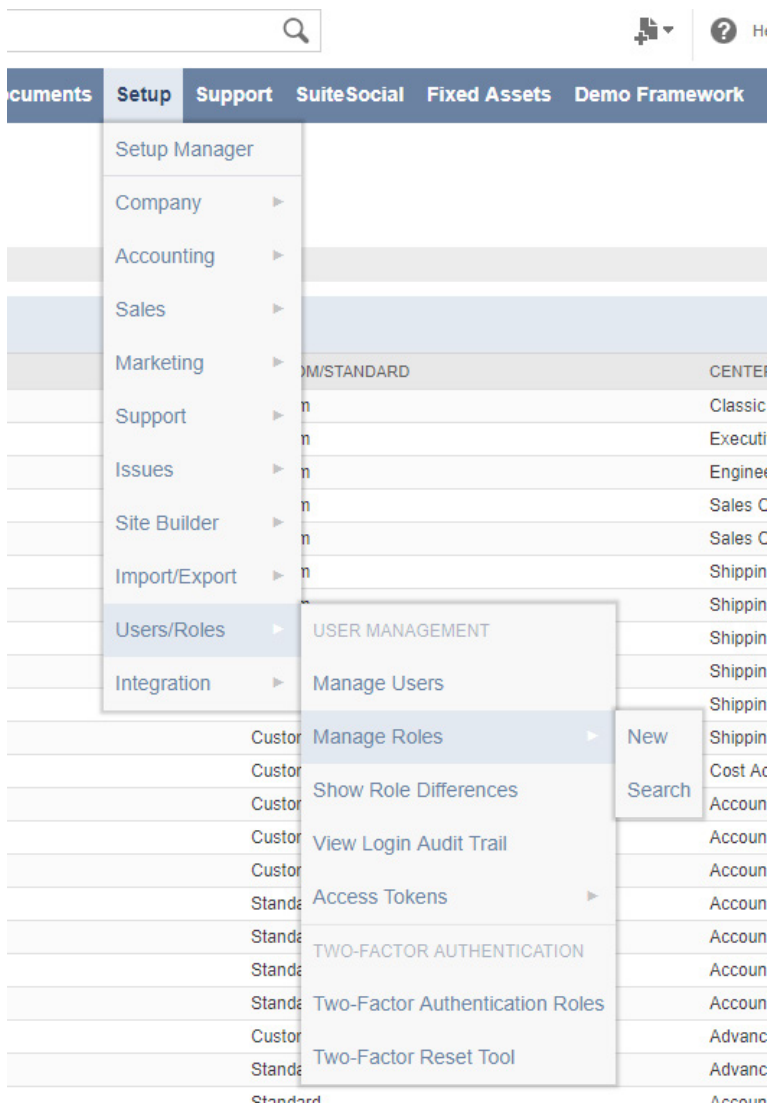
☒ SUITECLOUD DEVELOPMENT FRAMEWORK

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](#)

4. Once the change is saved, go to Setup > Users/Roles > Manage Roles



5. User should now see the Developer role. (This role won't be available until user enables SDF feature.) Click Customize

Edit	1039	Customer Service Rep	Custom	Support Center
Edit	1028	Customer Service Rep - Sales	Custom	Sales Center
Customize	55	Developer	Standard	Classic Center
Edit	1047	Ecommerce Manager	Custom	E-Commerce Management Center

6. In the role creation page, update the name to SDF Role. Set Access Subsidiaries to All.

Role

Save

Cancel

Reset

Change ID

General

NAME *

SDF Role

ID

CENTER TYPE
Classic Center

EMPLOYEE RESTRICTIONS

none - no default

☐ ALLOW VIEWING

ISSUE ROLE

Subsidiary Restrictions

ACCESSIBLE SUBSIDIARIES

☒ ALL

☐ ACTIVE

☐ USER SUBSIDIARY

☐ SELECTED

☐ ALLOW CROSS-SUBSIDIARY RECORD VIEWING

7. Add Find Transaction under Permissions > Transactions

Permissions	Restrictions	Forms	Searches	Users	Preferences	Dashboard	Translation	History
Transactions •	Reports •	Lists •	Setup •	Custom Record				
PERMISSION *							LEVEL	
Find Transaction							Full	
✓ Add	✕ Cancel	✚ Insert	🗑 Remove					

8. Add Publish Search under Permissions > Lists (If there is a custom search needs to be set as public for custom objects to use, this permission is required.)

Permissions		Restrictions	Forms	Searches	Preferences	Dashboard	Translation
Transactions Reports • Lists • Setup • Custom Record							
PERMISSION *	LEVEL						
Custom Record Entries	Full						
Documents and Files	Full						
Email Template	Full						
Marketing Template	Full						
PDF Template	Full						
Perform Search	Full						
<input type="button" value="✓ OK"/> <input type="button" value="✕ Cancel"/> <input type="button" value="+ Insert"/> <input type="button" value="🗑 Remove"/>							
Publish Search	Full						
Website (External) publisher	Full						
+ Add Row							

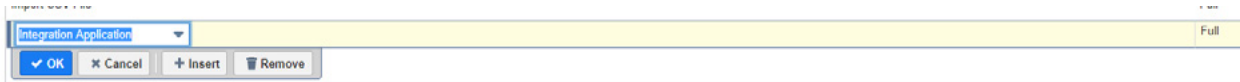
9. Remove Integration Applications from Permissions > Lists (Not needed for 2019.2)

Permissions		Restrictions	Forms	Searches	Preferences	Dashboard	Translation
Transactions Reports • Lists • Setup • Custom Record							
PERMISSION *	LEVEL						
Custom Record Entries	Full						
Documents and Files	Full						
Email Template	Full						
Integration Applications	Full						
<input type="button" value="✓ OK"/> <input type="button" value="✕ Cancel"/> <input type="button" value="+ Insert"/> <input type="button" value="🗑 Remove"/>							
Marketing Template	Full						
PDF Template	Full						
Perform Search	Full						
Website (External) publisher	Full						
+ Add Row							

10. Remove Access Token Management from Permissions > Setup (This option is only available when the Token-Based Authentication feature is enabled)(Not needed for 2019.2)

Permissions		Restrictions	Forms	Searches	Preferences	Dashboard	Translation
Transactions Reports • Lists • Setup • Custom Record							
PERMISSION *	LEVEL						
Access Token Management	Full						
<input type="button" value="✓ OK"/> <input type="button" value="✕ Cancel"/> <input type="button" value="+ Insert"/> <input type="button" value="🗑 Remove"/>							

11. Remove Integration Applications from Permissions > Setup (Not needed for 2019.2)

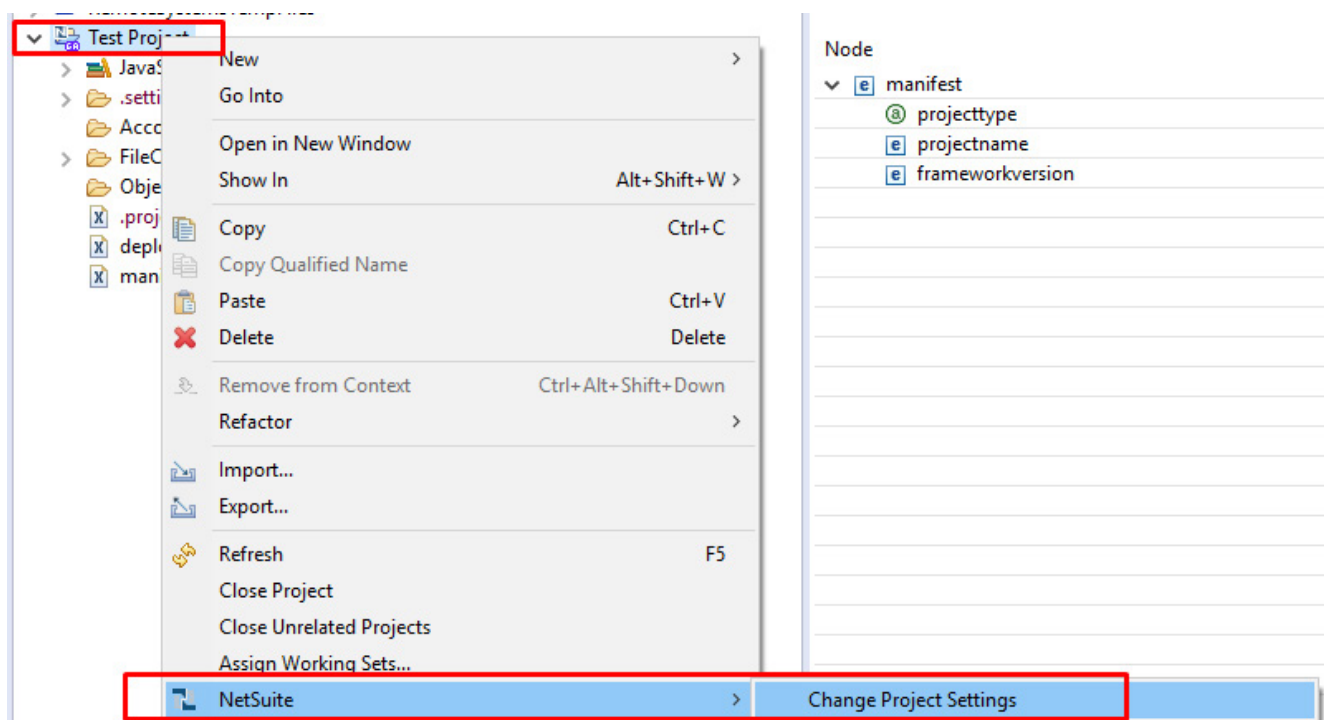


12. Save the Customized role.

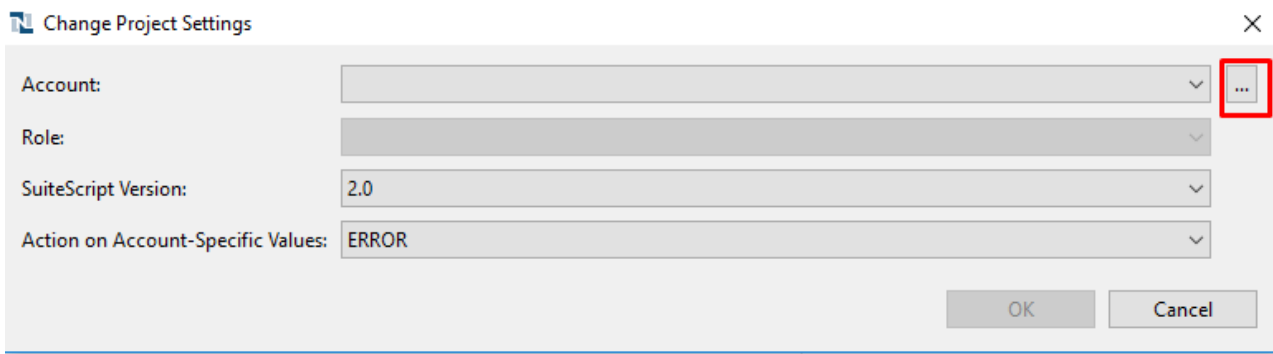
Configure Accounts for SDF Project in Eclipse

Change Project Settings

The 'Change Project Settings' menu option shown in the figure below is accessible from the SDF Project in Eclipse and allows the user to select which account and role combinations SDF will point to by default.



Right-click the parent project folder and navigate to NetSuite > Change Project Settings.



Change Project Settings

Account: ...

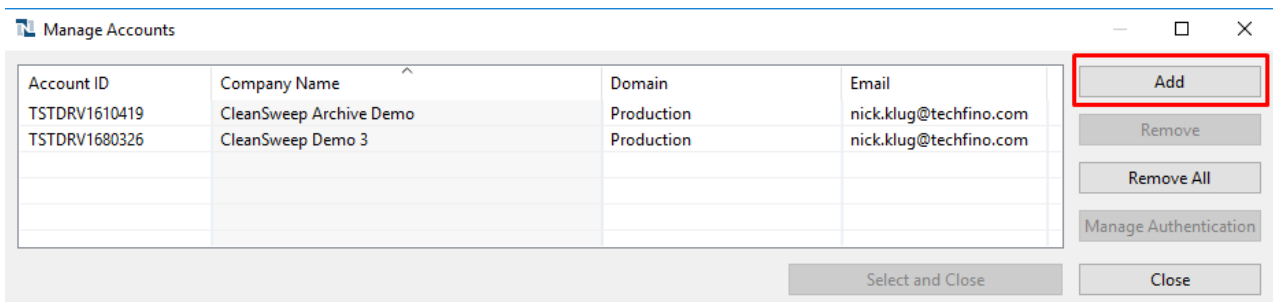
Role:

SuiteScript Version: 2.0

Action on Account-Specific Values: ERROR

OK Cancel

To add environments at roles to a project, click on the ellipsis in the top-right.

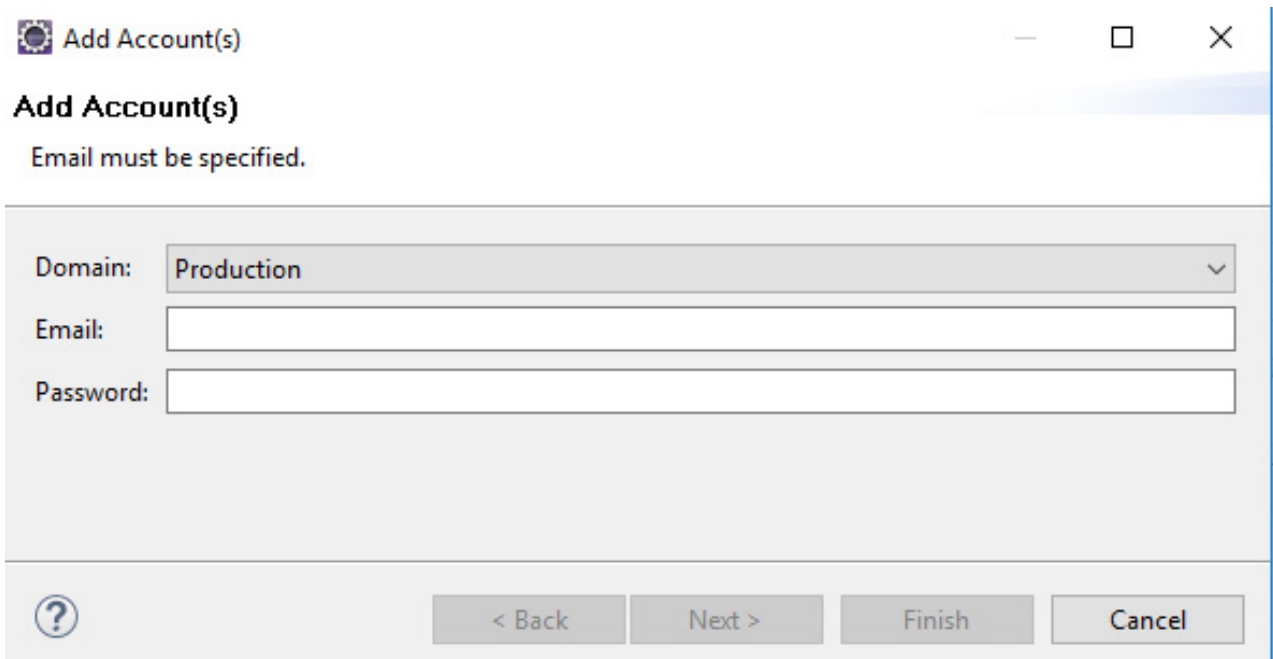


Manage Accounts

Account ID	Company Name	Domain	Email
TSTDRV1610419	CleanSweep Archive Demo	Production	nick.klug@techfino.com
TSTDRV1680326	CleanSweep Demo 3	Production	nick.klug@techfino.com

Add Remove Remove All Manage Authentication Select and Close Close

Select "Add" at the top right.



Add Account(s)

Email must be specified.

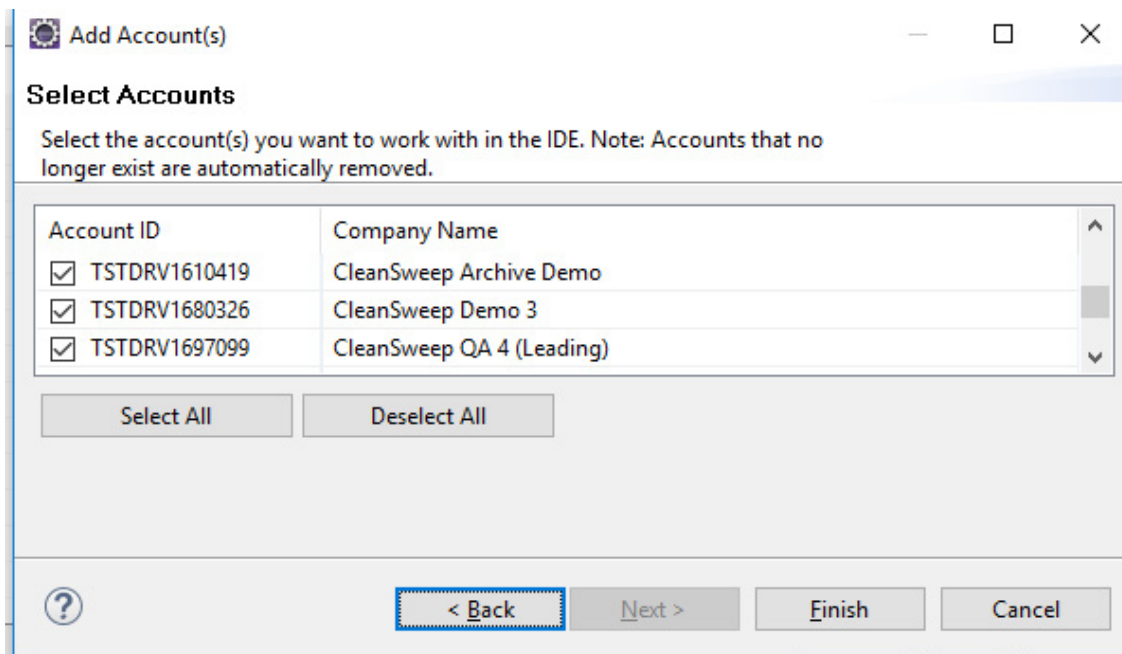
Domain: Production

Email:

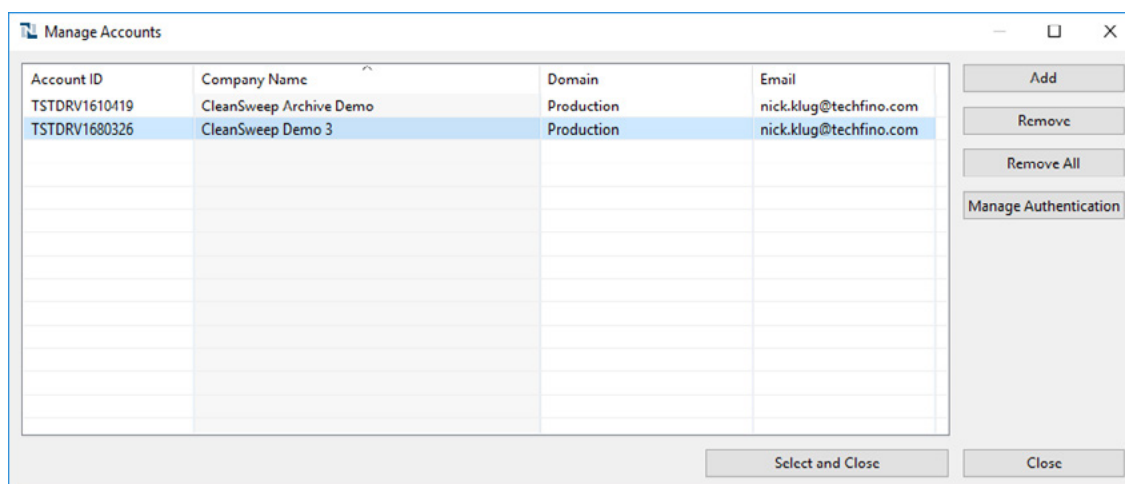
Password:

? < Back Next > Finish Cancel

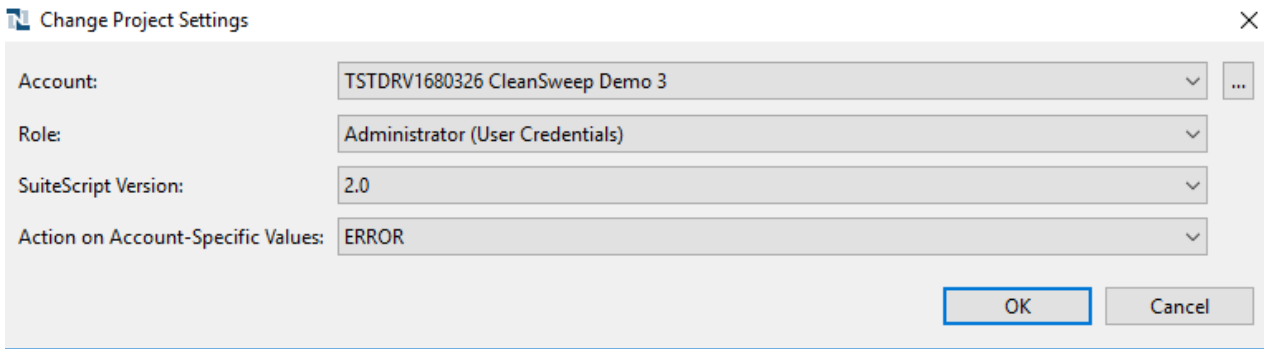
Now, select which environments you wish to add account logins for. With the reconfiguration of NetSuite's account specific logins, these will all be under "Production" or "Release Preview." Enter your credentials and click "Next."



You will then see a 'Manage Accounts' window allowing you to check all the accounts you wish to use your SDF project with.



Press "Select and Close" to select one of the environments as a default or "Close" to leave the dialog window.

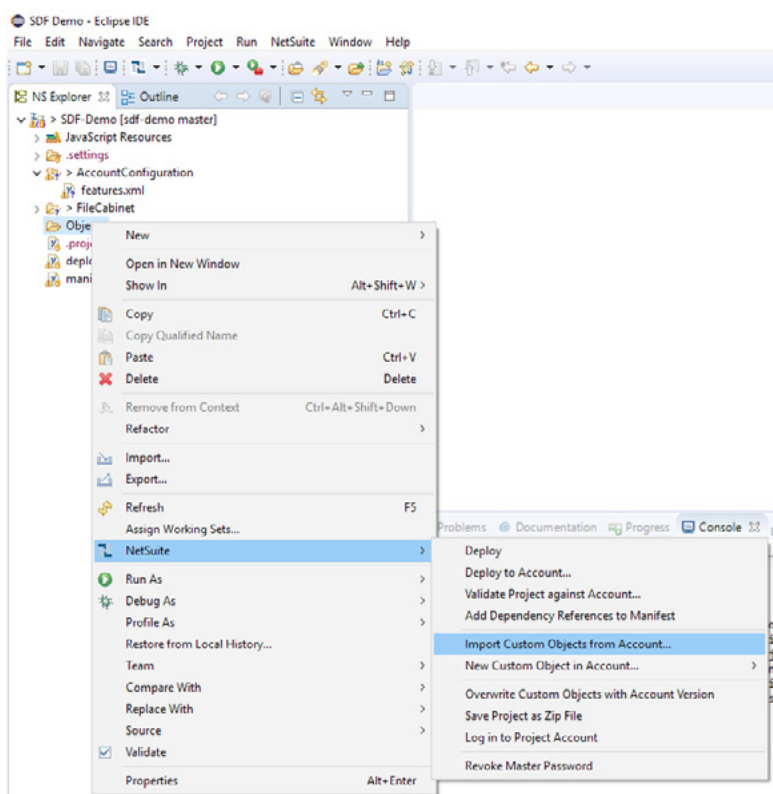


The 'Change Project Settings' window shown in the figure above allows the user to select which account and role combination SDF will default to for its operations. This process can be repeated at any point and it is recommended to start with it set to your Source environment as you develop, then switching to your target environment as you validate and deploy.

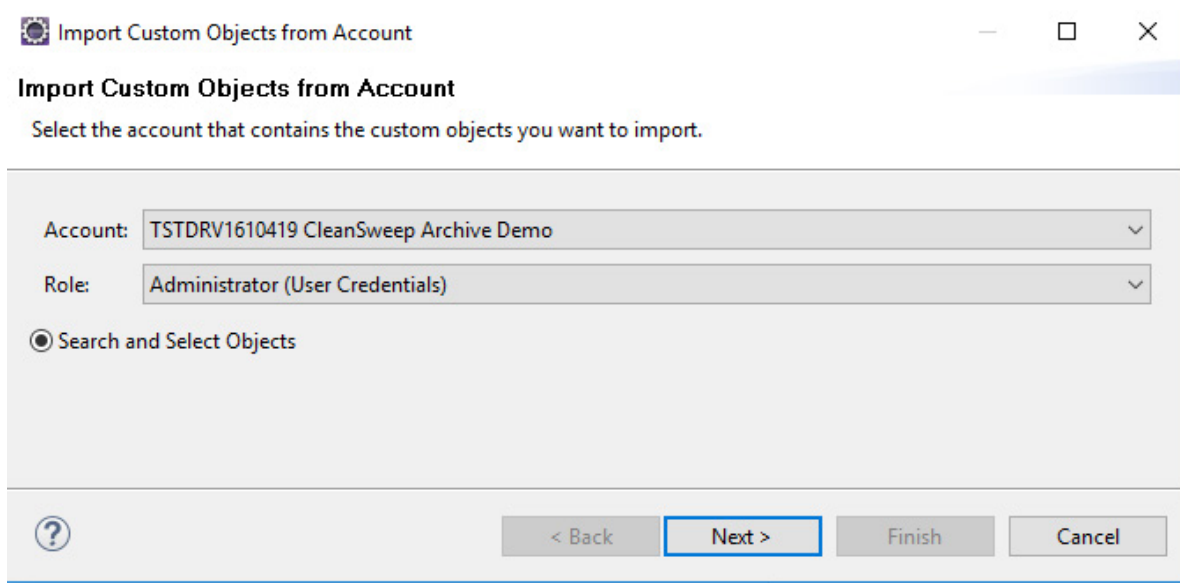
Add Customizations to SDF

Import by ScriptID

In this section, the flexibility to pull pre-existing customizations into your SDF project from a target NetSuite Account is on full display.



Right-click on the Objects folder in Eclipse and select "Import Custom Objects from Account."



Import Custom Objects from Account

Select the account that contains the custom objects you want to import.

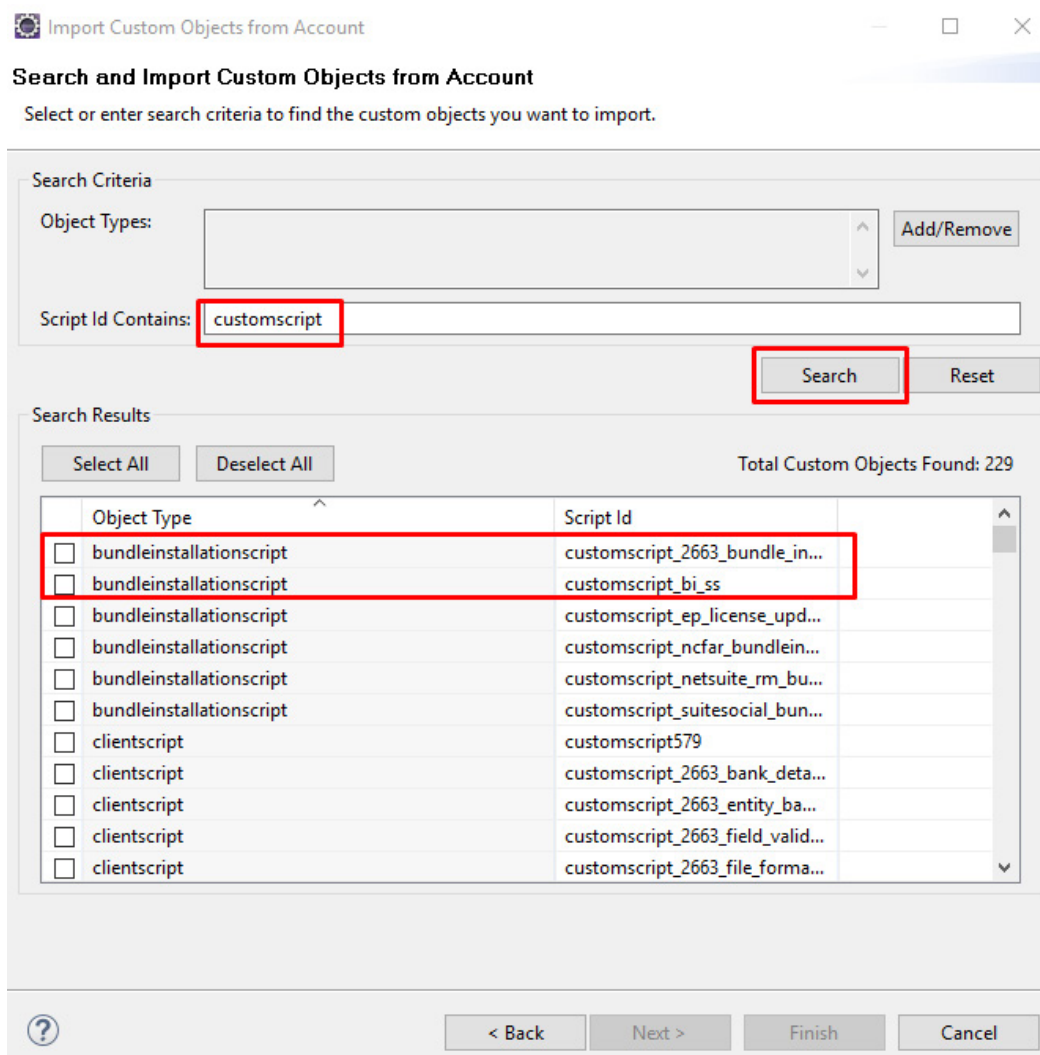
Account: TSTDRV1610419 CleanSweep Archive Demo

Role: Administrator (User Credentials)

☒ Search and Select Objects

< Back **Next >** Finish Cancel

Select the account and role needed to access the object, then click "Next"



Search and Import Custom Objects from Account

Select or enter search criteria to find the custom objects you want to import.

Search Criteria

Object Types: Add/Remove

Script Id Contains: customscript

Search Reset

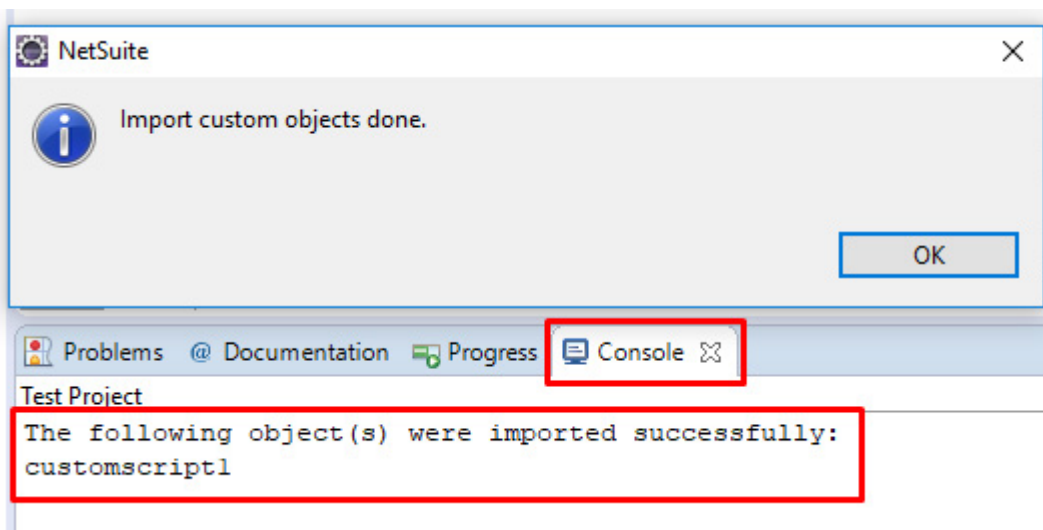
Search Results

Select All Deselect All Total Custom Objects Found: 229

Object Type	Script Id
<input checked="" type="checkbox"/> bundleinstallationscript	customscript_2663_bundle_in...
<input checked="" type="checkbox"/> bundleinstallationscript	customscript_bi_ss
<input type="checkbox"/> bundleinstallationscript	customscript_ep_license_upd...
<input type="checkbox"/> bundleinstallationscript	customscript_ncfar_bundlein...
<input type="checkbox"/> bundleinstallationscript	customscript_netsuite_rm_bu...
<input type="checkbox"/> bundleinstallationscript	customscript_suitesocial_bun...
<input type="checkbox"/> clientscript	customscript579
<input type="checkbox"/> clientscript	customscript_2663_bank_deta...
<input type="checkbox"/> clientscript	customscript_2663_entity_ba...
<input type="checkbox"/> clientscript	customscript_2663_field_valid...
<input type="checkbox"/> clientscript	customscript_2663_file_forma...

< Back Next > Finish Cancel

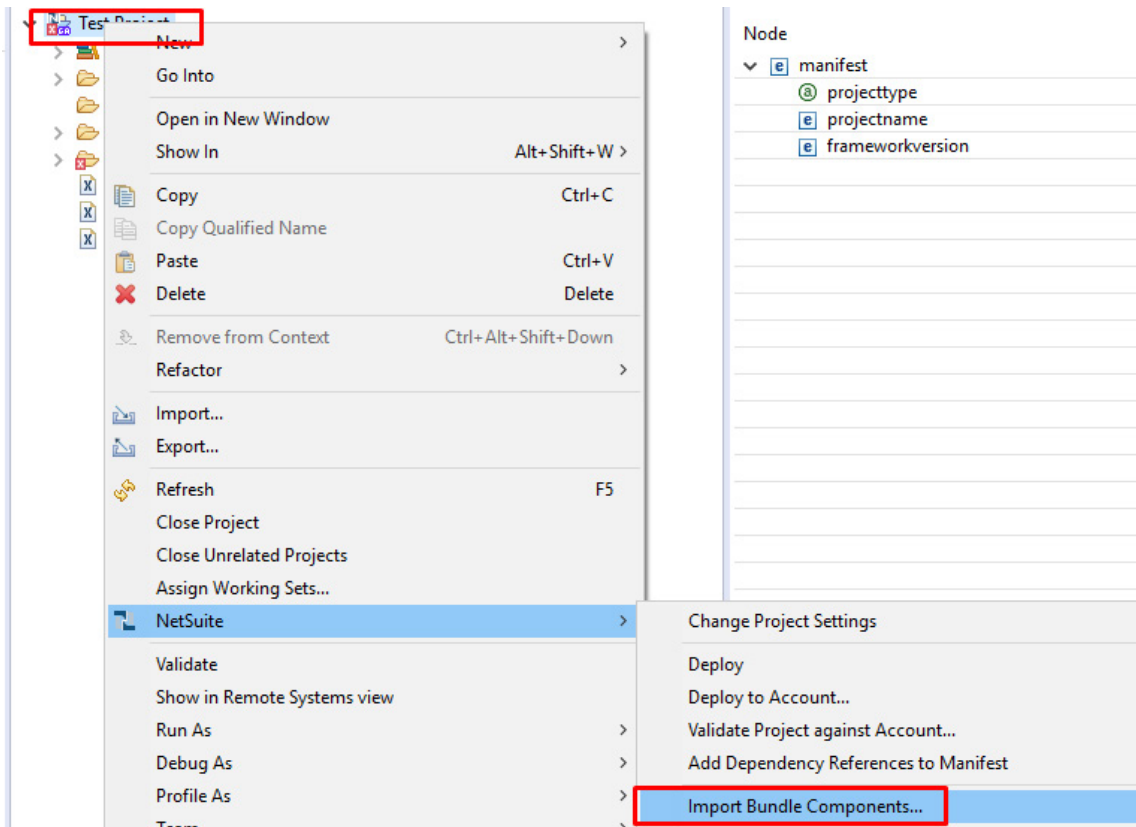
Enter the script ID in the above pane and then click "search." After the search finds all matches for the Script ID, check the checkboxes to select the objects you want to import into the project. Take careful note of the "Select All" and "Deselect All" buttons. Click "Finish" to begin importing the selected objects.



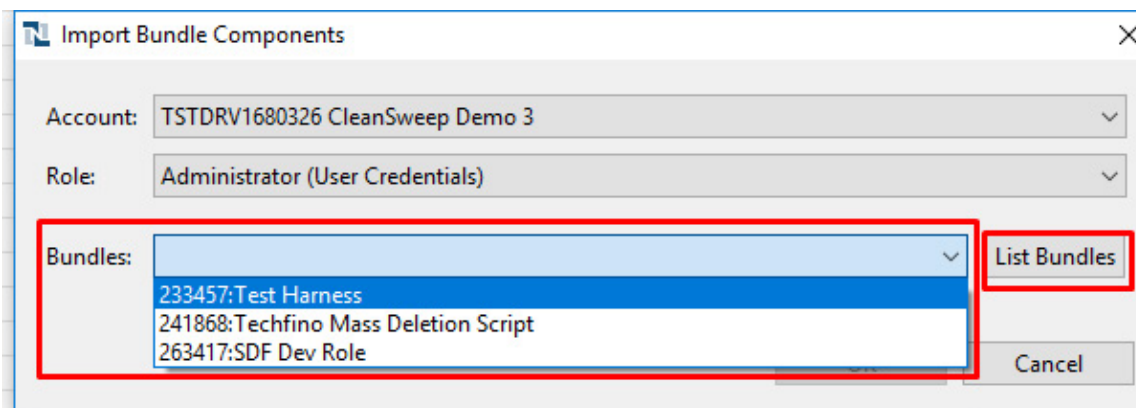
The objects will begin the import process. If there are any errors during the import process, the console will display error messages detailing why the import failed. The majority of the issues that occur during the import process are due to permissions. Ensure the role you are using to import has the proper permissions for the objects being imported.

Import by Bundle ID

Another nice feature is the ability to import objects into your project by SuiteBundle ID. This can be accomplished by right-clicking on the parent project name in Eclipse and select "Import Bundle Components."

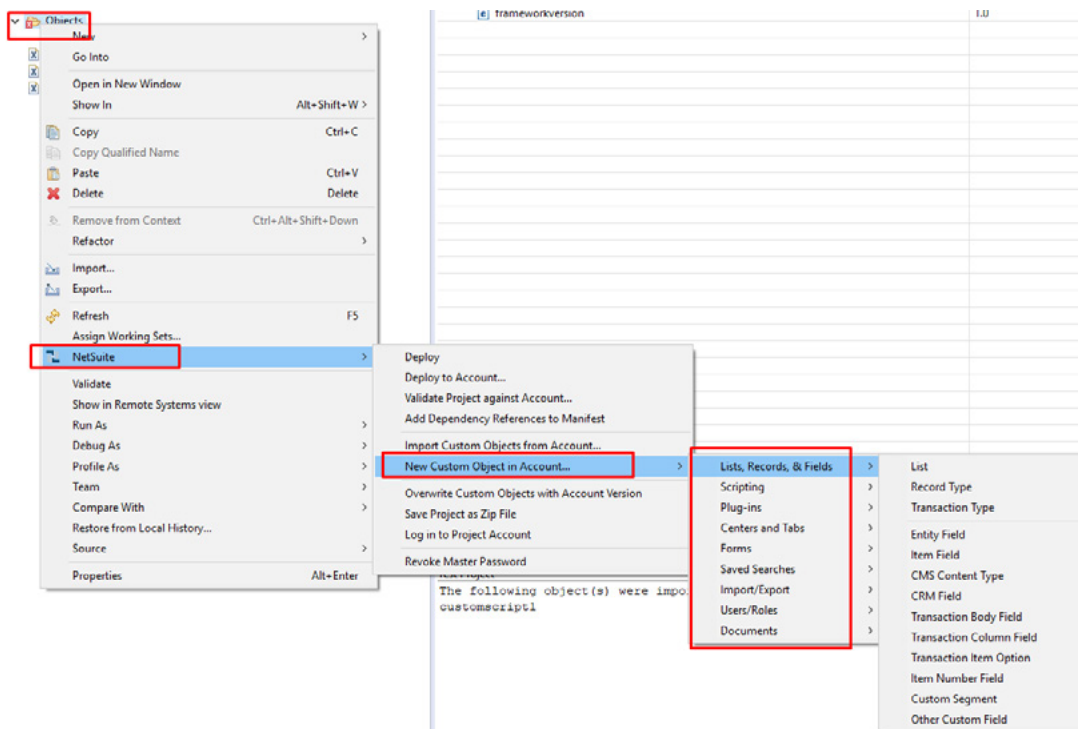


Another nice feature is the ability to import objects into your project by SuiteBundle ID. This can be accomplished by right-clicking on the parent project name in Eclipse and select "Import Bundle Components."



Next, select "List Bundles", select the bundle you wish to import into SDF, then click OK. SDF will automatically import the objects in the bundle into your SDF project.

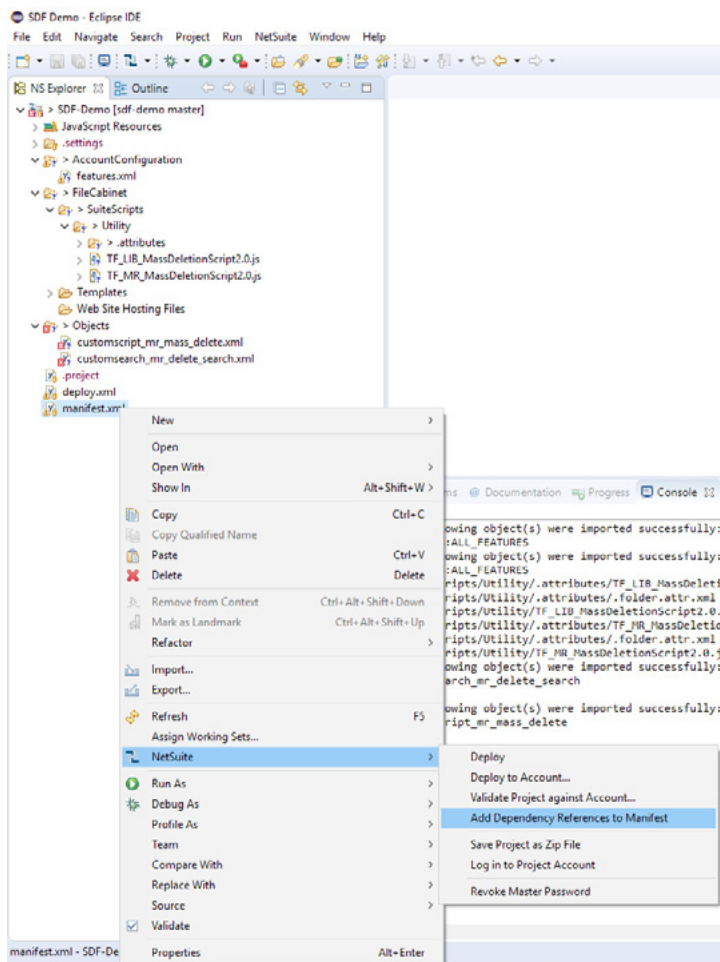
Manually Create Customizations



Right-click on the Objects folder in the SDF project. Navigate to NetSuite > New Object In Account, then select the customization object you want to add.

Adding Dependencies in SDF

With SuiteBundler, you often times run into deployment issues using SuiteBundles to deploy Customizations especially when they automatically include referenced objects from other SuiteBundles. This leads to some SuiteBundles including fields or records from other SuiteApps or SuiteBundles unintentionally because they were referenced in a Workflow for example. But this is one of the areas where SDF provides greater control and flexibility to either automatically include referenced objects or not in your Project.



Updating the manifest can be done two ways. The simplest method is to right-click anywhere in the project, navigate to NetSuite > Add Dependency References to Manifest. This will add all required dependencies for the objects currently in the project and add them to the project.

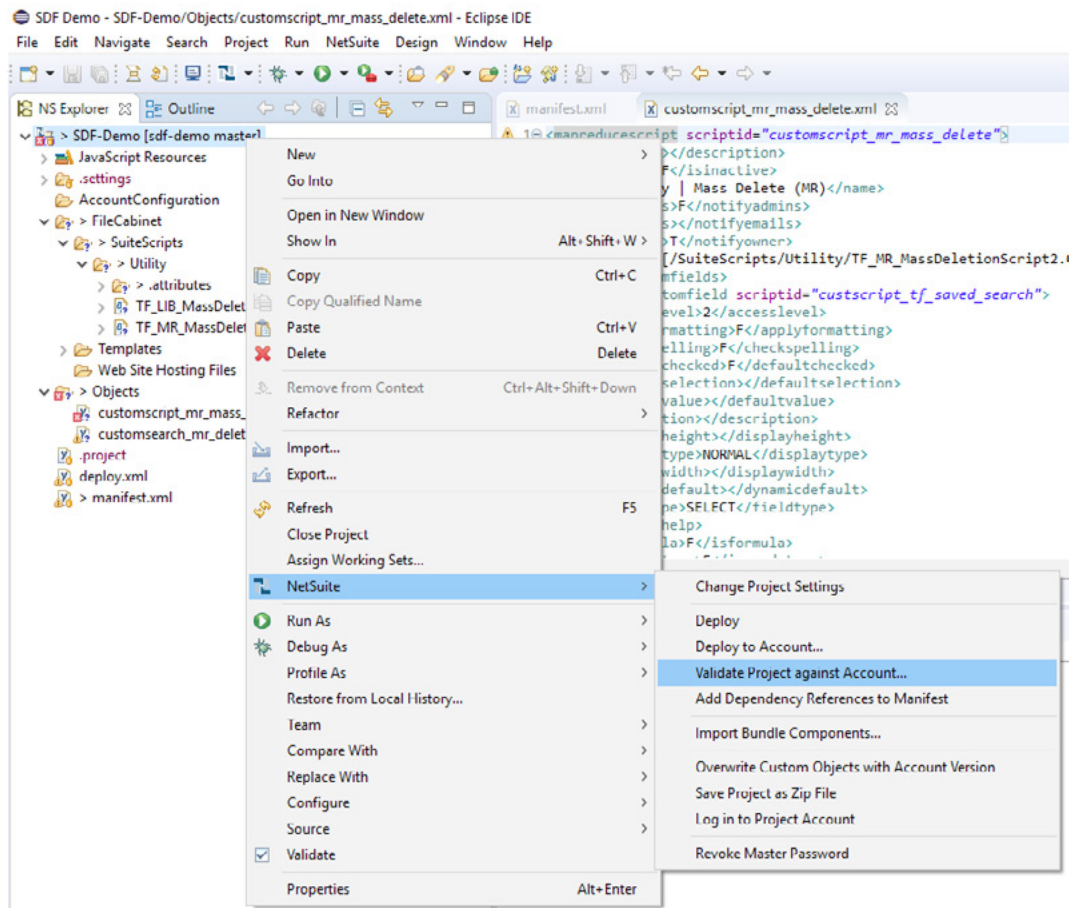
Alternatively, you can manually add dependencies to the project for the objects you know you will be working with as you go.


For the first few SDF projects, we highly recommend using the automatic addition of dependencies and paring back as you discover objects that you need to be included for the project to deploy.

Validation and Deploying SDF Project

Validate Project Against Account

Before deploying your SDF Project to a target account you can first validate the Project has all the prerequisites for installation. To do so, right-click anywhere in the project, navigate to NetSuite > Validate Project Against Account.



 Validate Project Against Account ✕

Account:

TSTDRV1610419 CleanSweep Archive Demo

Role:

Administrator (User Credentials)

Action on Account-Specific Values:

ERROR

Validate

Cancel

Next, select the account and role that you wish to use in order to validate the SDF project. Click validate and the process will begin.

```
Test Project
Validate account settings    Failed
*** ERROR ***

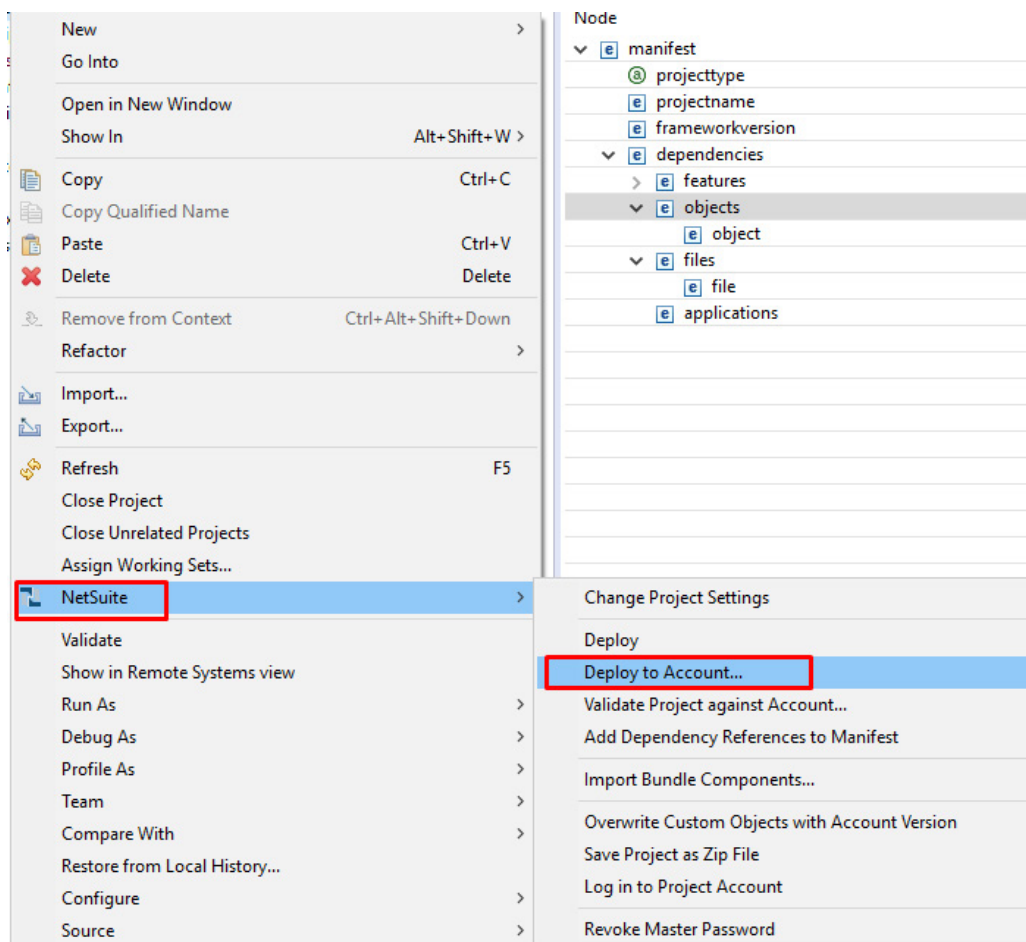
Validation of account settings failed.

An error occurred during account settings validation.
Details: The manifest contains a dependency on customrecord_product_review object, but it is not in the account.
Details: The manifest contains a dependency on /SuiteScripts/linkReviewToItem_v3.js file, but it is not in the account.
```

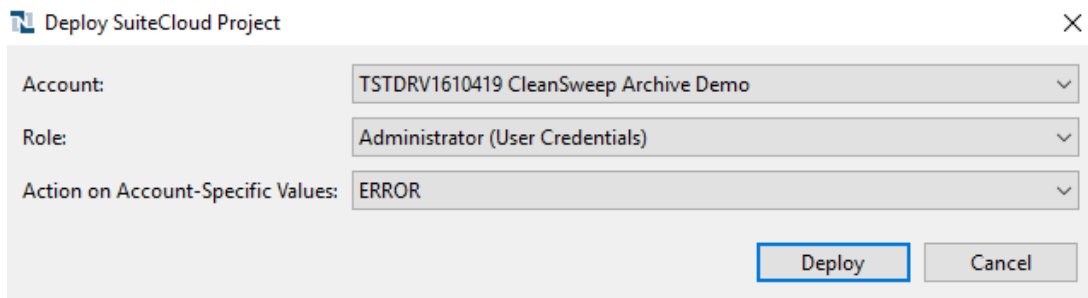
The validation process will check against the account to see if the project contains the settings and objects necessary to support the deployment - for example, if your SDF project requires Server Side Scripting in the manifest, and the target account does not have it, it will throw the related error. If the SDF project contains a reference in the manifest to customrecord_product_review, then customrecord_product_review needs to already be present in the target account. If it is not present, you will receive a warning similar to the above.

Deploy SDF Project to Account

After successfully validating your project against the target account, you are then set to perform the deployment. To initiate the deployment start by right-clicking anywhere in the project and navigate to NetSuite > Deploy to Account.



Next, select the account and role for project deployment. Click Deploy, and the process will begin. If the target Account is a Production environment, then there will be an additional confirmation window as shown below:

A screenshot of a 'Deploy SuiteCloud Project' dialog box. The dialog has a title bar with a close button (X). It contains three dropdown menus: 'Account' with the value 'TSTD RV1610419 CleanSweep Archive Demo', 'Role' with the value 'Administrator (User Credentials)', and 'Action on Account-Specific Values' with the value 'ERROR'. At the bottom right, there are two buttons: 'Deploy' and 'Cancel'.

If the project deployment is not successful, you will see error messages similar to what was contained in the validation section. Sometimes, there are additional issues in an environment that cannot be caught during the validation stage. This typically relates to data conflicts in the target Account. For example, duplicate script IDs, values used in things such as a script parameter or workflows that are not contained in the target Account.

Sample: Mass Delete Script Deployment

Background

Previously, we posted [NetSuite Mass Delete Tool](#), developed and documented by Luke. Today, we are going to use this solution for a quick example for using SDF as a deployment tool. Below are the steps for what's involved in the customization.

- **Mass Delete Search** - A saved search created to identify the records to be deleted.
 - In this case, we don't actually want to delete any transactions. As a result, we added Date Created after today as a criteria for this search
 - Search columns doesn't really matter. We kept as is and didn't change anything
 - The search must be set to public to be allowed to be selected under script deployment

Saved Transaction Search

List Search Copy to Account (Beta) More Export

Mass Delete Search

Save & Run **Reset** **Cancel** **Preview** **New Template** **Pivot Report** **Change ID** **Actions**

SEARCH TITLE *

Mass Delete Search

ID

customsearch_mr_delete_search

OWNER

RenHong Zhu

☒ 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 Tra**

Use this tab to specify criteria that narrow down your search.

☐ USE EXPRESSIONS

Standard • **Summary**

FILTER *	DESCRIPTION *	FORMULA
Main Line	is true	
Date Created	is on today	
Type	is Journal	

Add **Cancel** **Insert** **Remove**

Save & Run **Reset** **Cancel** **Preview** **New Template** **Pivot Report** **Change ID** **Actions**

Criteria **Results** **Highlighting** **Available Filters** **Audience** **Roles** **Email** **Audit Trail** **Execution Log**

Use this tab to indicate columns to be included in the search results as well as sort order

SORT BY

Order Type

☐ DESCENDING

THEN BY

☐ DESCENDING

THEN BY

☐ DESCENDING

OUTPUT TYPE

Normal

CONSOLIDATED EXCHANGE RATE

Per-Account

☐ SHOW TOTALS

MAX RESULTS

☐ RUN UNRESTRICTED

☐ MY PREFERRED SEARCH RESULTS

Columns • **Drill Down Fields**

Remove all

Add Multiple

FIELD *	SUMMARY TYPE	FUNCTION	FORMULA	WHEN ORDERED BY FIELD	CUSTOM LABEL	CUSTOM LABEL TRANSLATION
Order Type						
*						
Date						
As Of Date						
Period						
Tax Period						
Type						
Document Number						
Name						
Account						
Memo						
Amount						

Add **Cancel** **Insert** **Remove** **Move Up** **Move Down** **Move To Top**

- **Utility | Mass Delete (MR)** – The Map/Reduce script record

Script

[Edit](#)
[Back](#)
[Deploy Script](#)
[Actions ▾](#)

TYPE

Map/Reduce

NAME

[Utility | Mass Delete \(MR\)](#)

ID

customscript_mr_mass_delete

API VERSION

2.0

[Scripts](#)
[Parameters](#)
[Unhandled Errors](#)
[Execution Log](#)
[Deployments](#)
[System Notes](#)

SCRIPT FILE

preview TF_MR_MassDeletionScript2.0.js [download](#) [Edit](#)

☒ GET INPUT DATA

☒ MAP

☒ REDUCE

☒ SUMMARIZE

Custom Plug-In Types

CUSTOM PLUG-IN TYPE

No records to show.

- Script parameter: Saved Search for Delete – this parameter is passed to the script to identify what is going to be deleted

Scripts Parameters Unhandled Errors Execution Log Deployments System Notes			
New Parameter			
DESCRIPTION	ID	TYPE	LIST/RECORD
Saved Search for Delete	custscript_tf_saved_search	List/Record	Saved Search

[Edit](#)
[Back](#)
[Deploy Script](#)
[Actions ▾](#)

- **Script Deployment**
 - Set Parameter value to Mass Delete Search

Script Deployment

[Edit](#)
[Back](#)
[Actions ▾](#)

SCRIPT
Utility | Mass Delete (MR)

TITLE
Utility | Mass Delete (MR)

ID
customdeploy_mr_mass_delete

☒ DEPLOYED

STATUS
Not Scheduled

SEE INSTANCES
[Status Page](#)

LOG LEVEL
Debug

EXECUTE AS ROLE
Administrator

PRIORITY
Standard

CONCURRENCY LIMIT
2

☐ SUBMIT ALL STAGES AT ONCE

YIELD AFTER MINUTES
10

BUFFER SIZE
1

[Schedule](#) • [Parameters](#) • [Execution Log](#) • [System Notes](#)

SAVED SEARCH FOR DELETE
Mass Delete Search

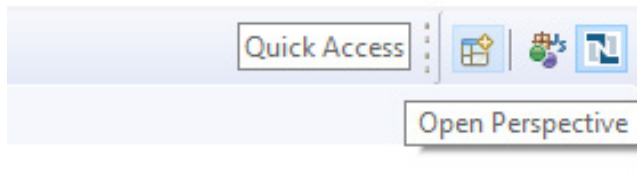
We are going to use Eclipse for this example to pull above customization from our CleanSweep Archives & Purge Dev and deploy to QA account. We will show you each step, hope this will be useful to you. We are assuming you already have Eclipse and NetSuite setup according to our SDF Development Environment Setup section which includes custom role setup, assigned to user and SDF plugin for Eclipse. This example is using Eclipse IDE 2019-03 R for Javascript and Web Developers edition.

Project and Git setup in Eclipse

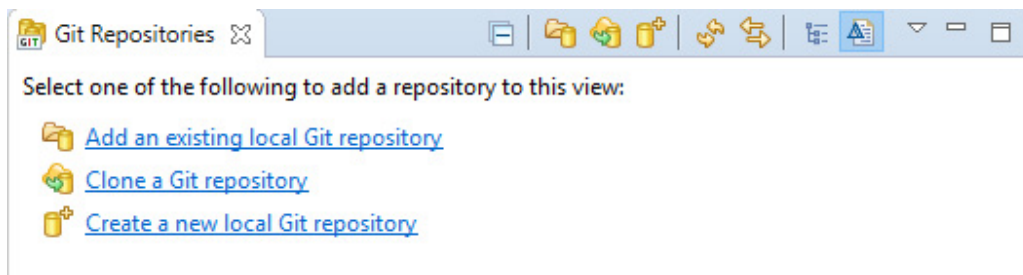
For demonstration purposes, we are using Eclipse git plugin for all repository interaction because it doesn't require any other installations. You can use your choice of repository and tool (SVN, GitKraken, etc.). An empty git repository sdf-demo has also been created via bitbucket for this example.

Add Git Perspective and Add Repository

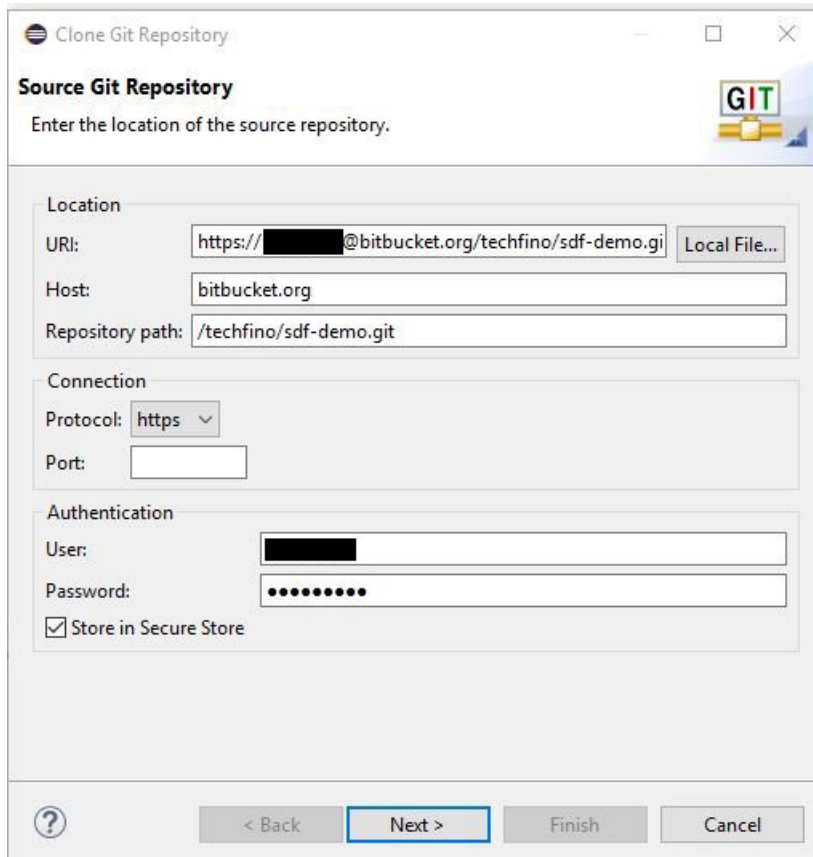
1. Click on Open Perspective located at the top right corner



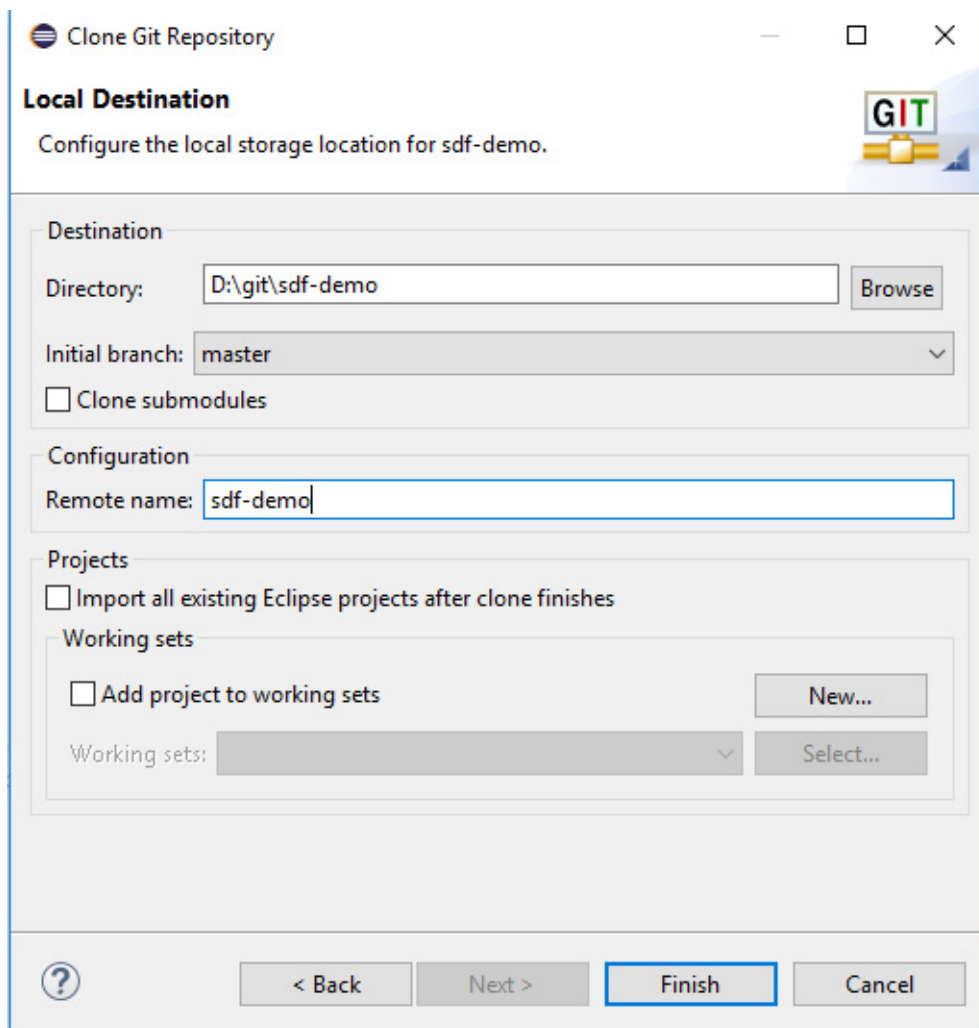
2. Select Git from the list and Click Open. Now the Git Perspective is opened, you should be able to see Git Repository section (Shown below)



3. Click on Clone a Git repository, enter the URI and required information and click next



4. Setup path and click Finish. Please be aware sdf-demo here is the name of the repository instead of SDF-Demo which is the name of the project.

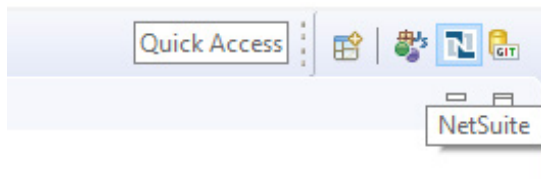


5. Now the bitbucket repository is added to your repository list

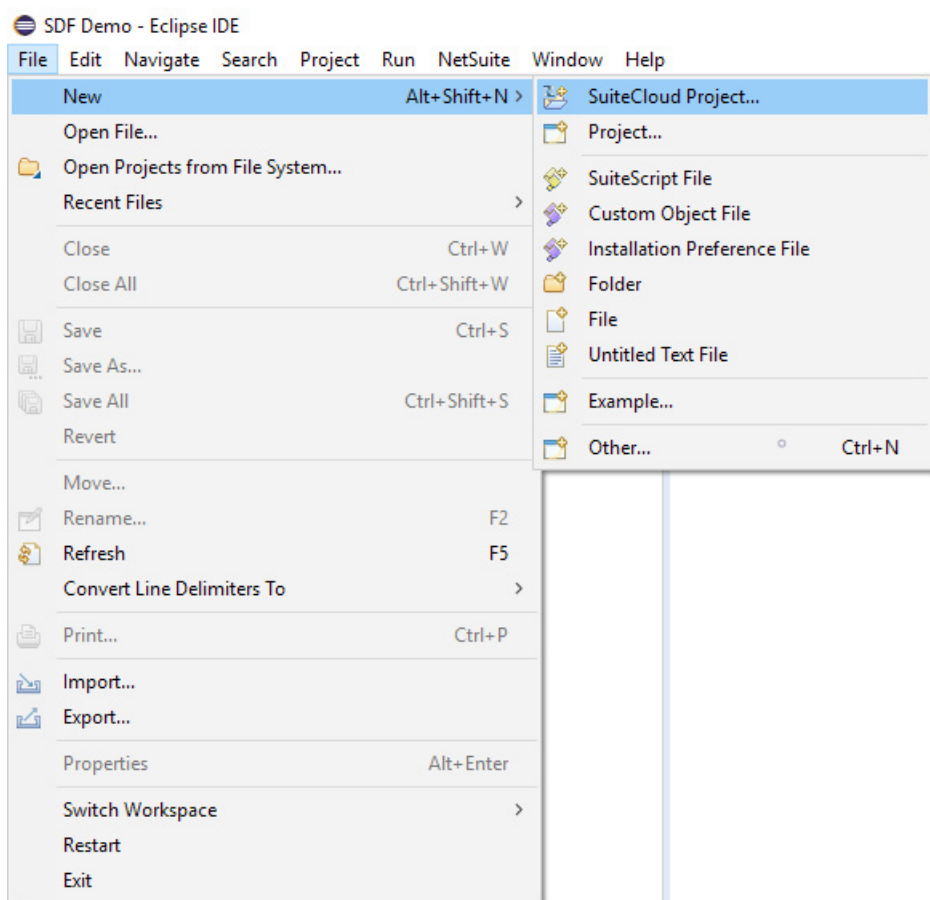


Create New SuiteCloud Customization Project

1. Click on SuiteCloud perspective at the top right corner

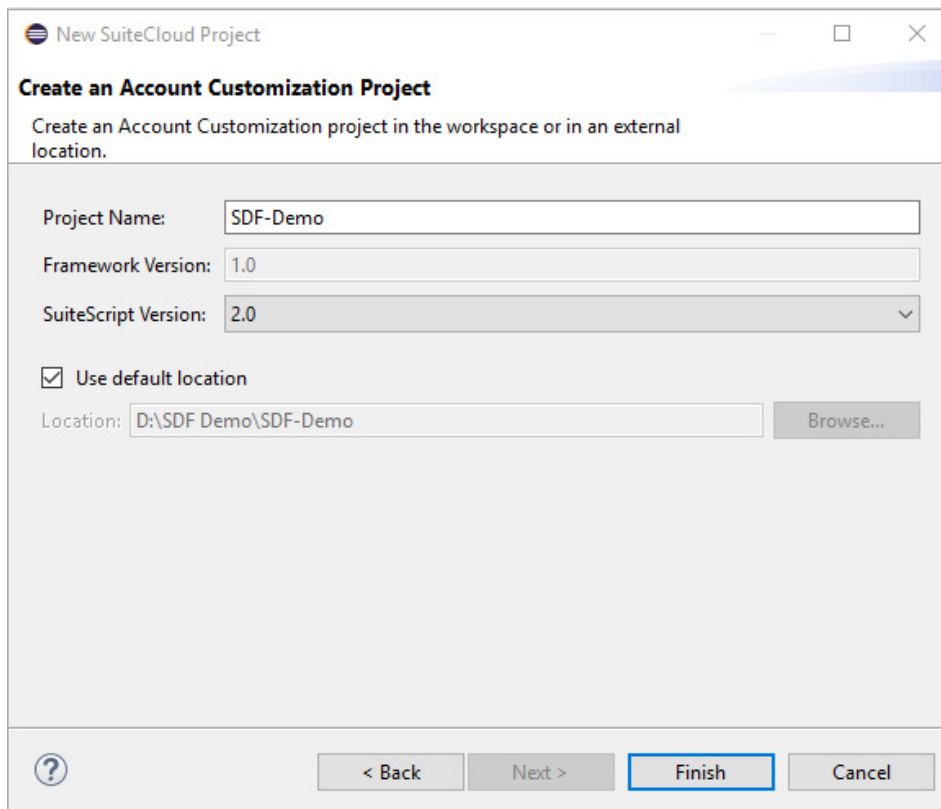


2. Go to File > New > SuiteCloud Project...



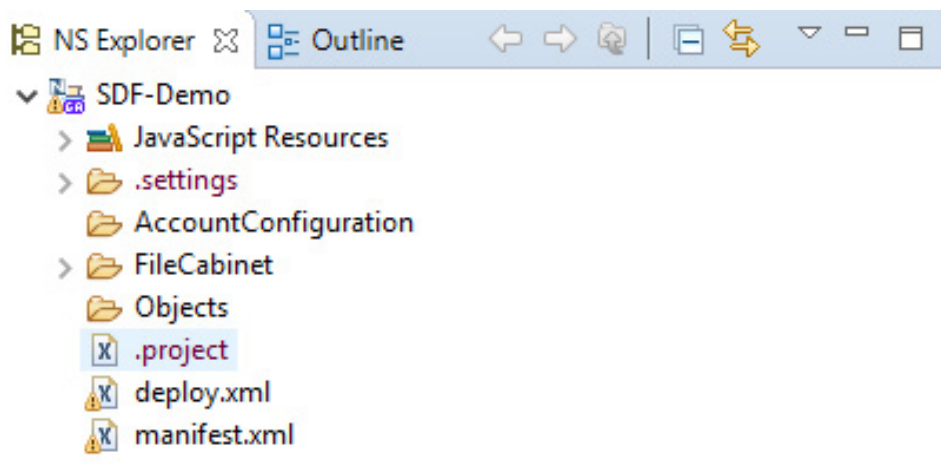
3. Select Account Customization and click Next

4. Enter Project Name: SDF-Demo and Click Finish



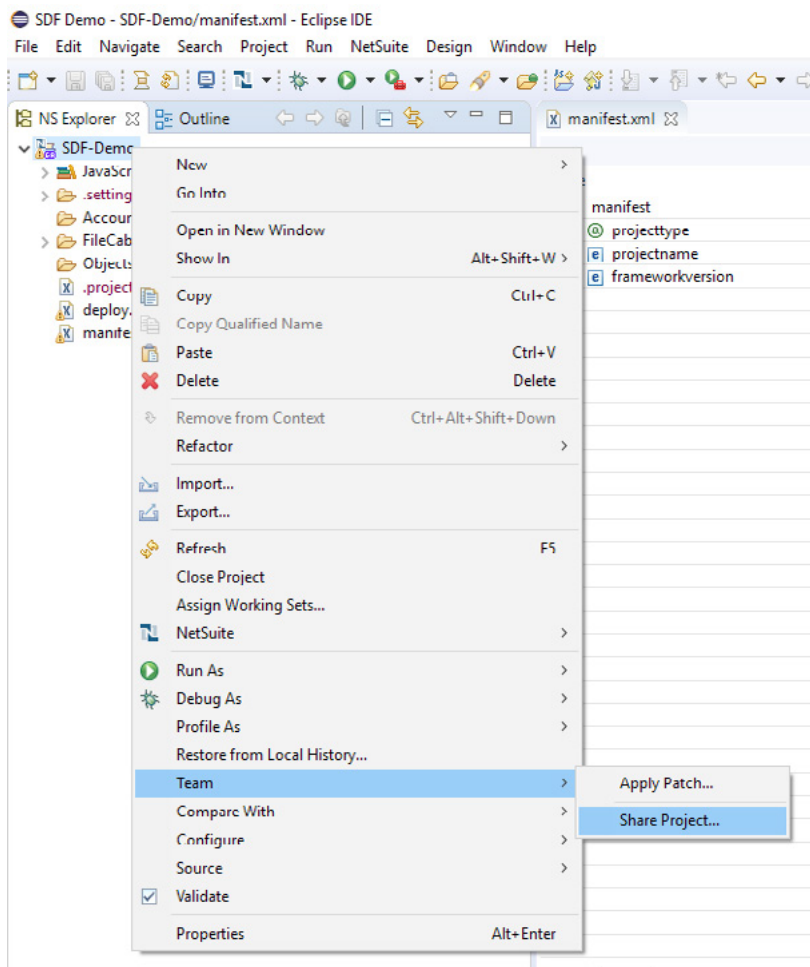
The screenshot shows a 'New SuiteCloud Project' dialog box. The title bar says 'New SuiteCloud Project'. The main heading is 'Create an Account Customization Project'. Below this, it says 'Create an Account Customization project in the workspace or in an external location.' The dialog has several input fields: 'Project Name' with the value 'SDF-Demo', 'Framework Version' with the value '1.0', and 'SuiteScript Version' with the value '2.0'. There is a checkbox labeled 'Use default location' which is checked. Below this, the 'Location' field shows 'D:\SDF Demo\SDF-Demo' and a 'Browse...' button. At the bottom, there are four buttons: '< Back', 'Next >', 'Finish' (which is highlighted with a blue border), and 'Cancel'.

5. You will see your new SuiteCloud project in NS Explorer

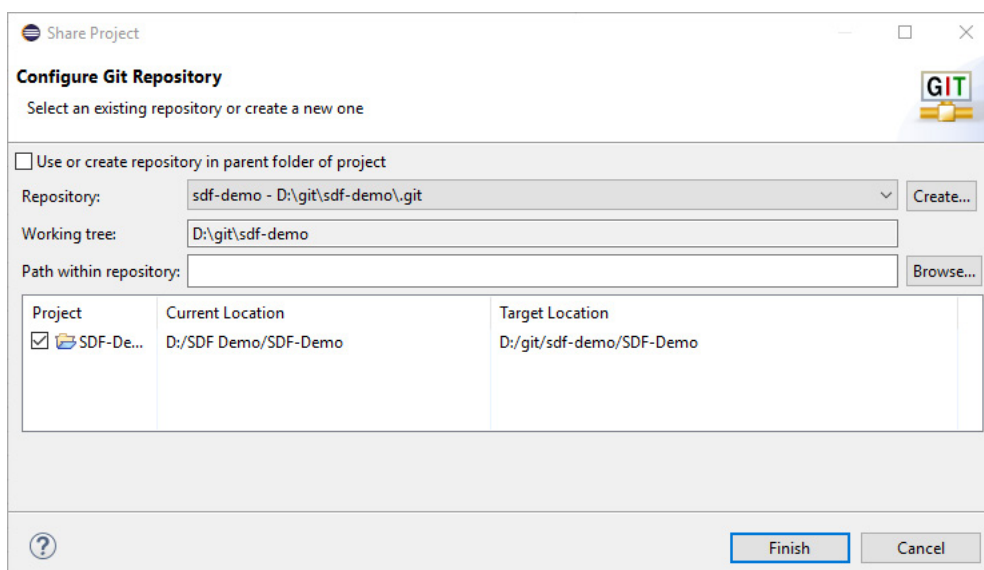


Link SuiteCloud Project to Repository

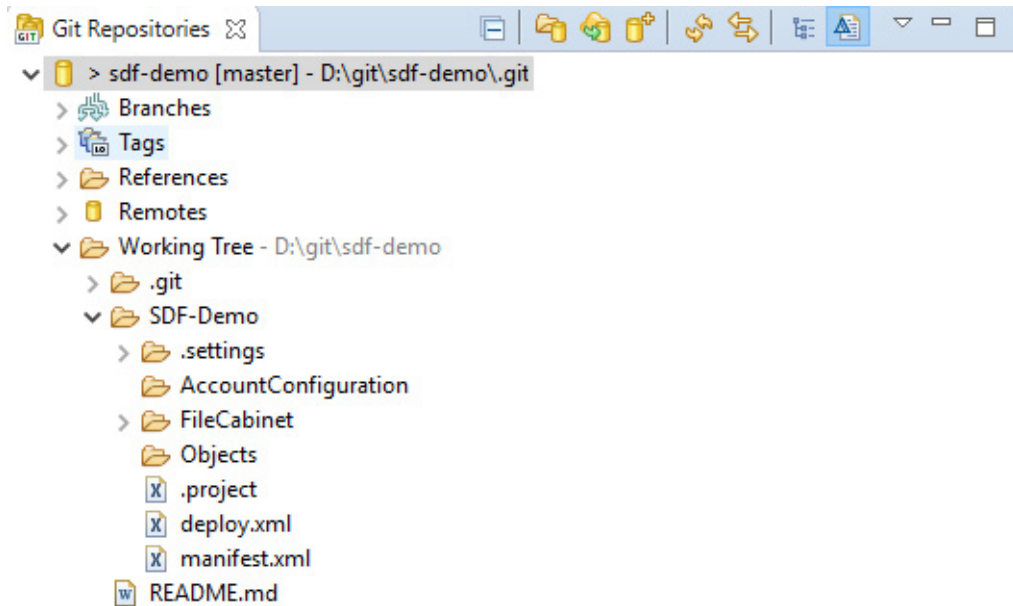
1. Right Click on SDF-Demo project and find Team > Share Project



2. Select sdf-demo under Repository field and click Finish



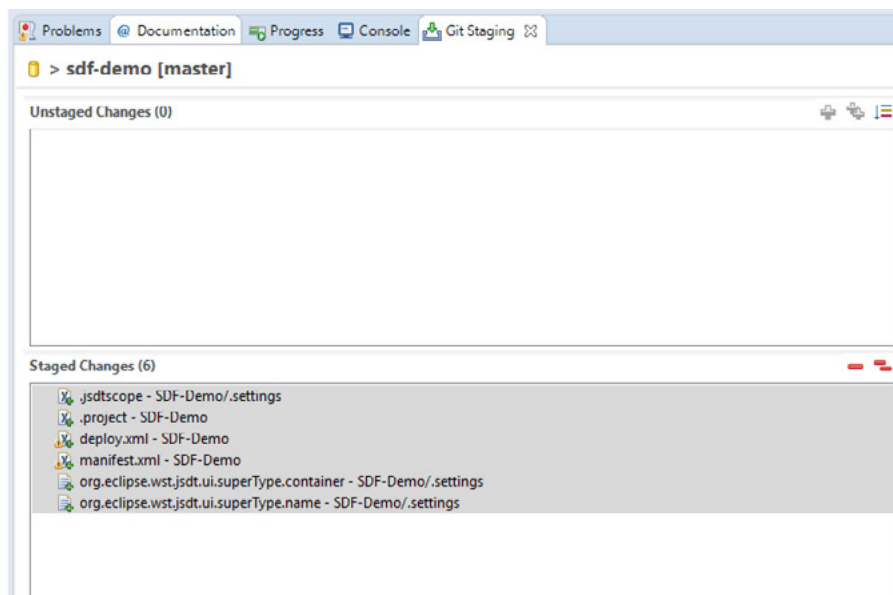
3. Now the Project is linked to our sdf-demo repository. If you go to Git perspective and expand sdf-memo repository. You will be able to find all the folders and files there.



Commit Empty SuiteCloud Project

At this point, we can commit and push the empty project to Git. It would help us to identify what are changed in later stage.

1. Right click on the SDF-Demo project and go to Team > Commit, the move all files to Staged Changes

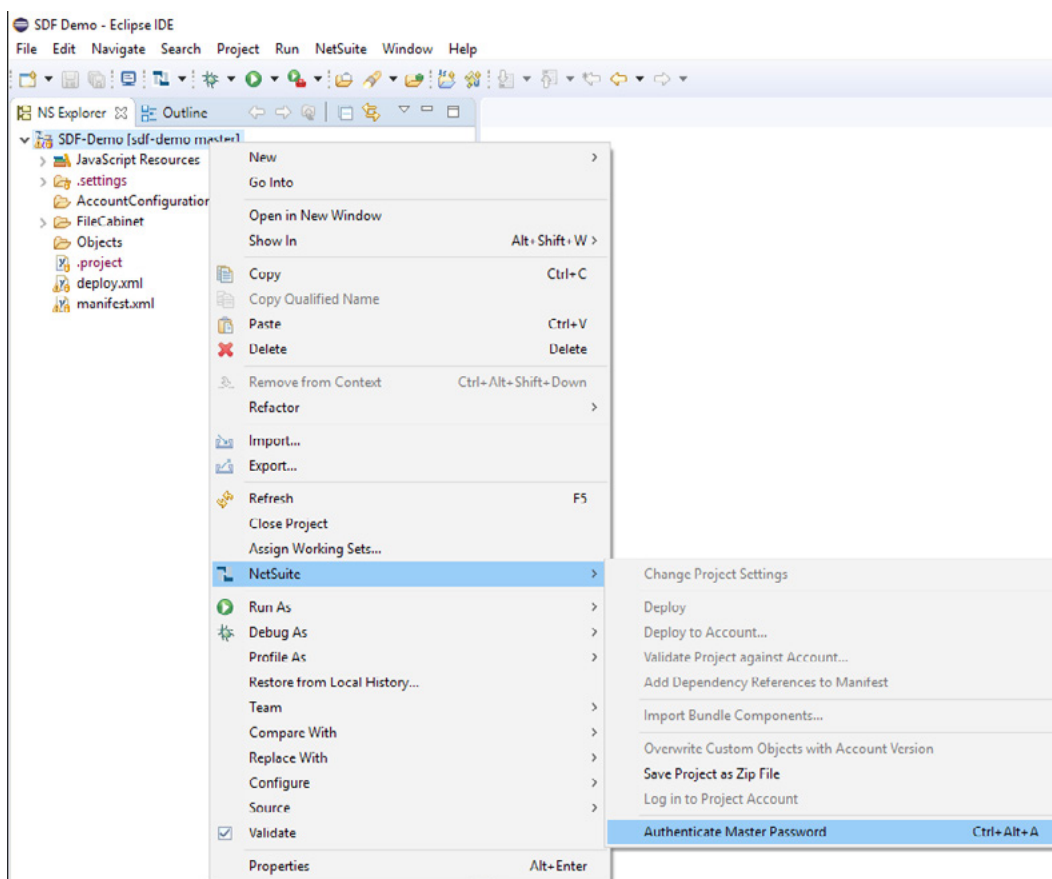


2. Enter Commit Message and click Commit and Push
3. Now the empty SuiteCloud project has been committed and pushed to our bit-bucket hosted git repository

Setup SuiteCloud Project Access

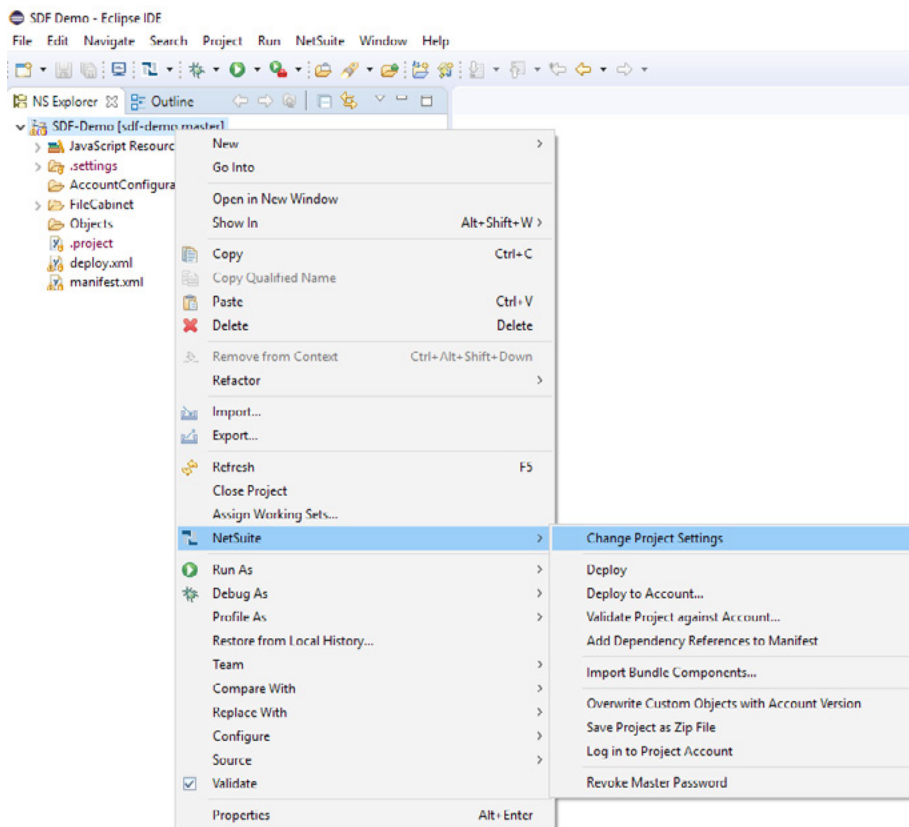
Now we need to set up the project access.

1. Right click SDF-Demo project and go to NetSuite > Authenticate Master Password and enter your master password. This password is the one you entered when you installed the SDF plugin for the first time.

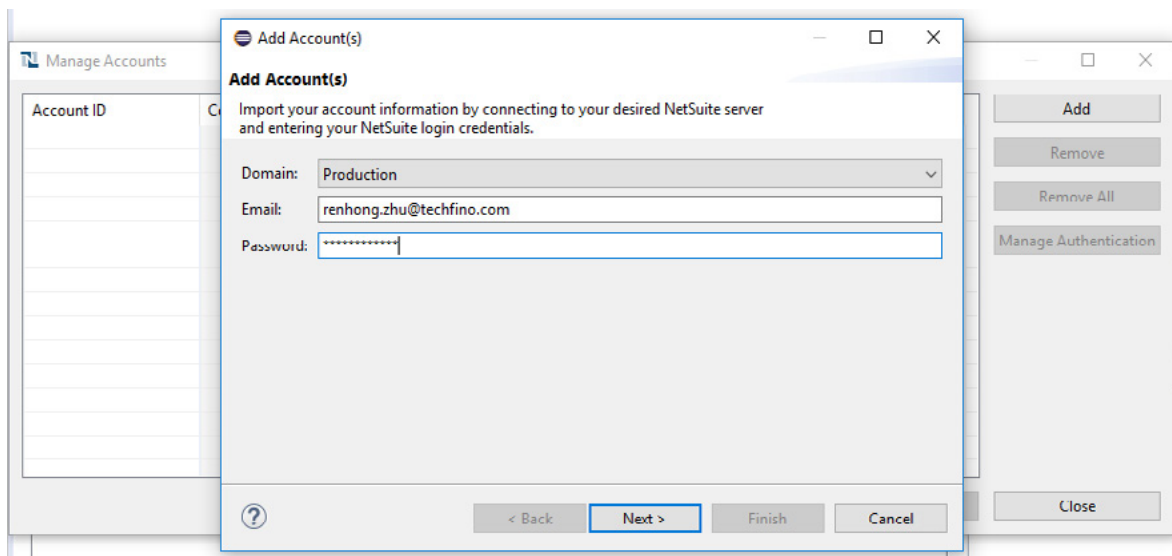


2. Once entered, all NetSuite related functions are enabled.

3. Right Click on the project and find NetSuite > Change Project Settings



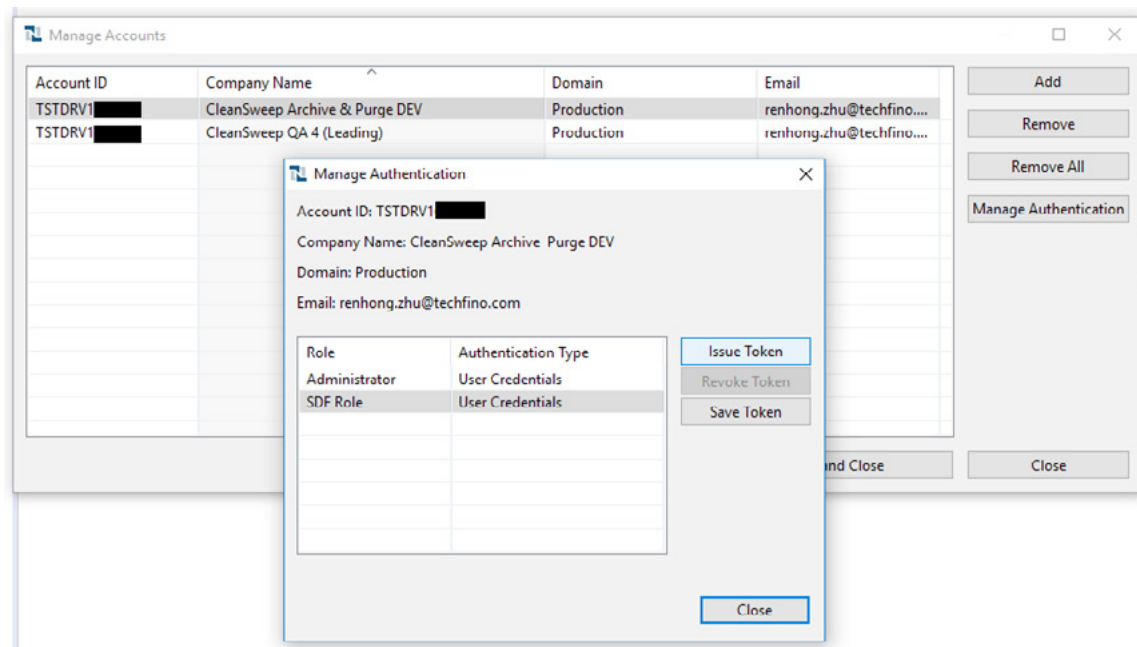
4. Click Add, enter your NetSuite username and password, then click Next



5. If the credential you entered is correct, you will see a list of account you have access to in the next page. In this case, we are moving customization from CleanSweep Archives & Purge Dev to QA. We are selecting those 2 accounts and Click finish.

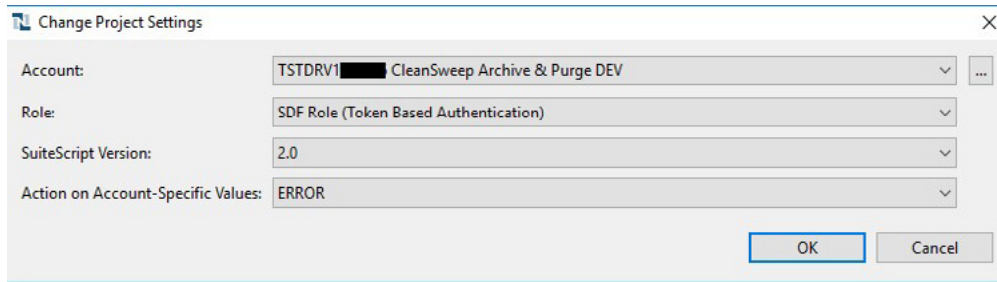
<input checked="" type="checkbox"/>	TSTDRV1 [REDACTED]	CleanSweep Archive & Purge DEV
<input checked="" type="checkbox"/>	TSTDRV1 [REDACTED]	CleanSweep QA 4 (Leading)

6. Now we need to setup Token access using SDF Role. Otherwise, NetSuite will kick you out of your UI session every time you import a customization object. Click on the first account in the list and click Manage Authentication on the right side. Then click Issue Token. You should see Token Issued messaged after a couple seconds. Do the same for the other account.



7. Next, select Dev account and Click Select and Close. In the future, when we need to import data, the Dev account will be the default account.

8. On the next page, change role to SDF Role

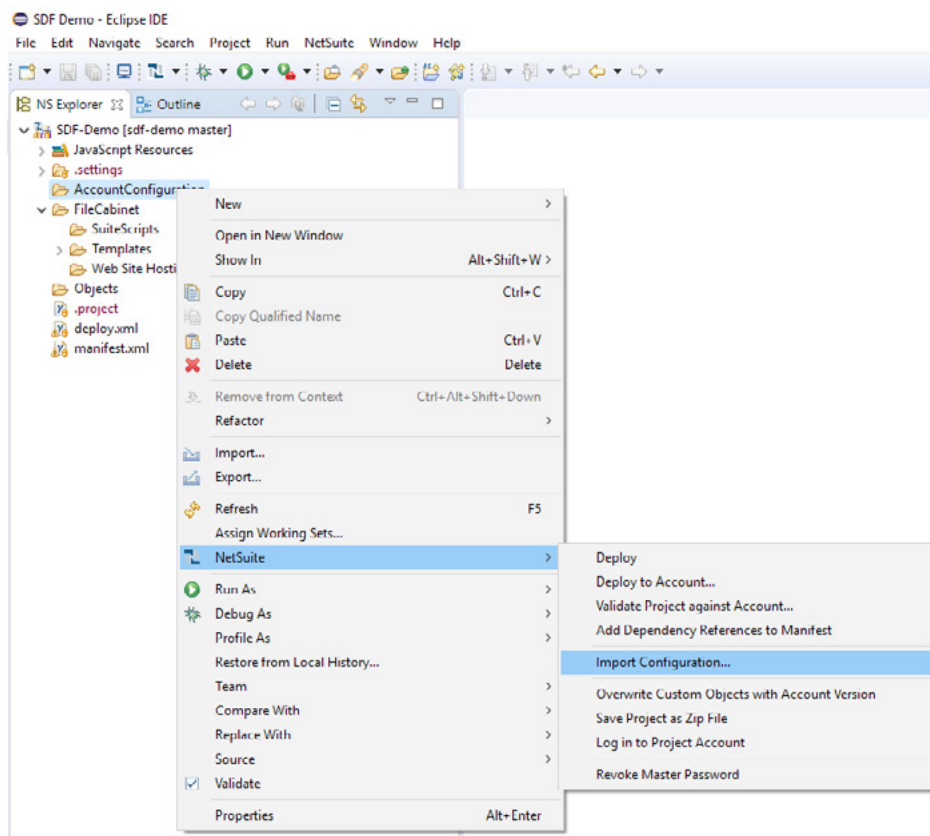


At this Point, access has been captured in Eclipse. For the next project, if you are using the same accounts, you don't have to do this again.

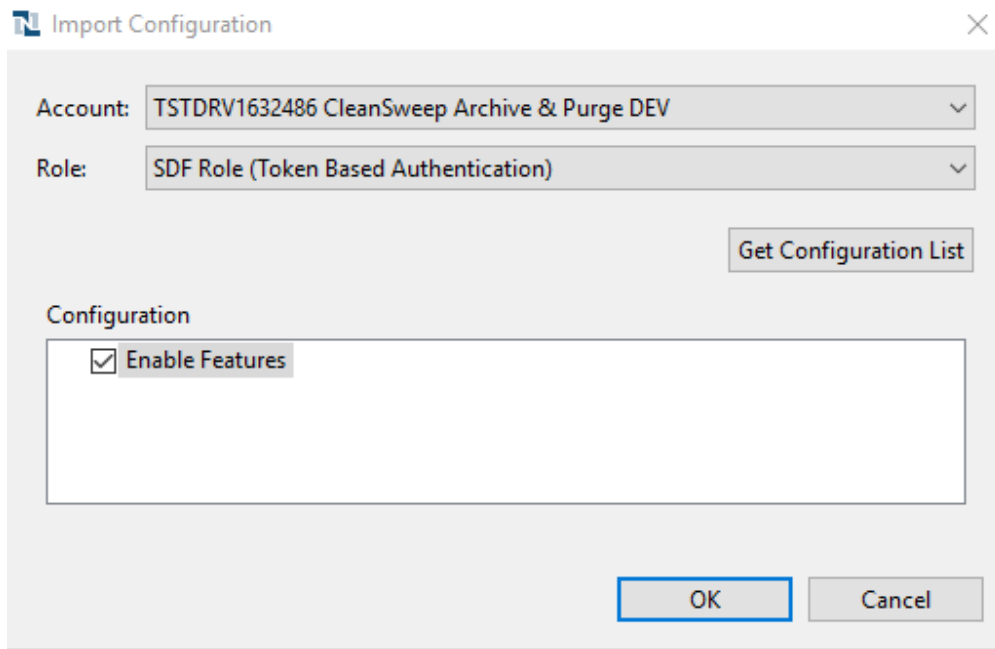
Import Customizations and Validate before Deployment

Finally it's time to import customization from Dev to Eclipse. Keep in mind, the script contains two parts. The script record under Objects folder and the actual script file under File Cabinet > SuiteScripts folder along with all the libraries.

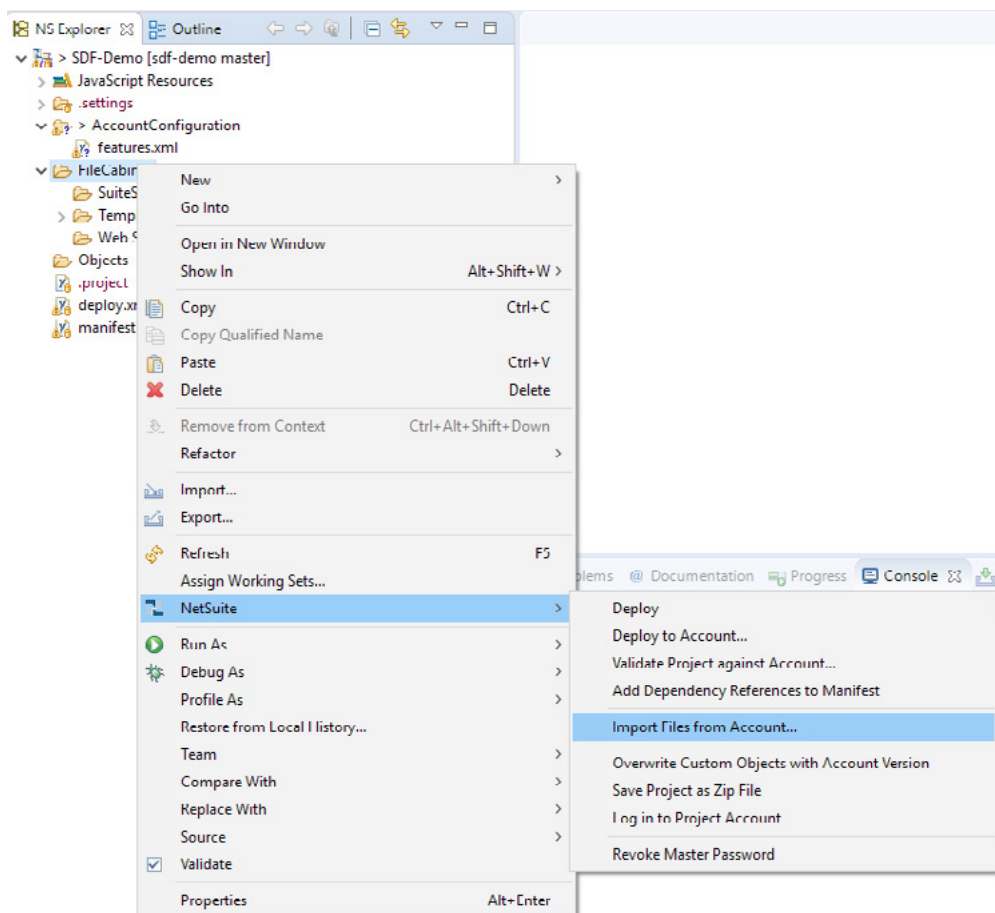
1. Right click the AccountConfiguration and find NetSuite > Import Configuration



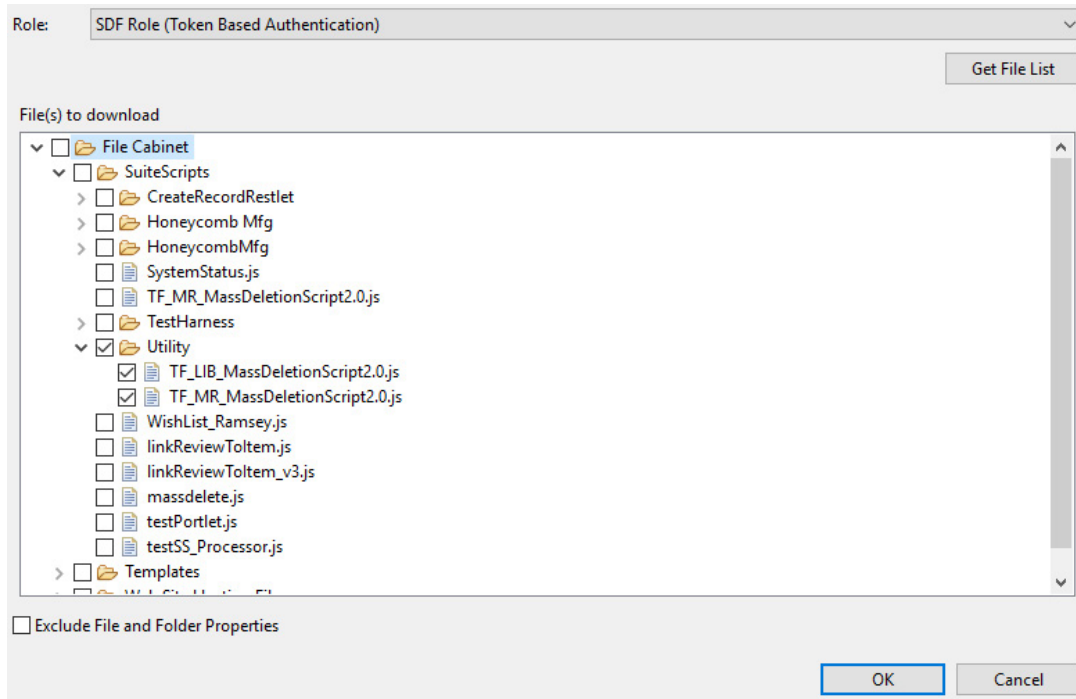
2. Click Get Configuration List, select Enable Features then OK



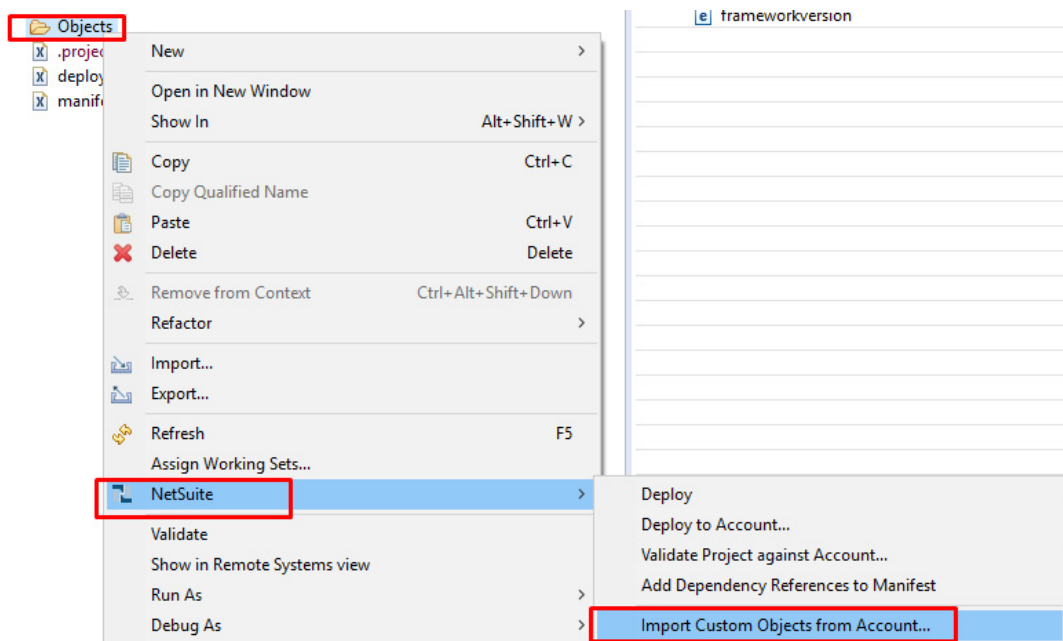
3. Right click the FileCabinet and find NetSuite > Import Files from Account



- Click Get File List and select the target folder. In this case, we only need files from the Utility folder. Click OK. Then NetSuite will start importing the script files from File Cabinet. Once done, you should see the script files under FileCabinet folder. Also, it's important to pay attention to the logs in the console. Should an error occur, you will see details highlighted in red. You will want to pay attention to logs whenever you import or deploy anything.



- Import the Saved Search object by right clicking Objects folder and go to NetSuite > Import Custom Objects from Account



- Click Next, put the NetSuite id for saved search in Script ID Contains field and click search. Find the check the correct search in the list and click Finish. Once imported, you should see the search object under Objects folder

Import Custom Objects from Account

Search and Import Custom Objects from Account

Select or enter search criteria to find the custom objects you want to import.

Search Criteria

Object Types: **Add/Remove**

Script Id Contains:

Search **Reset**

Search Results

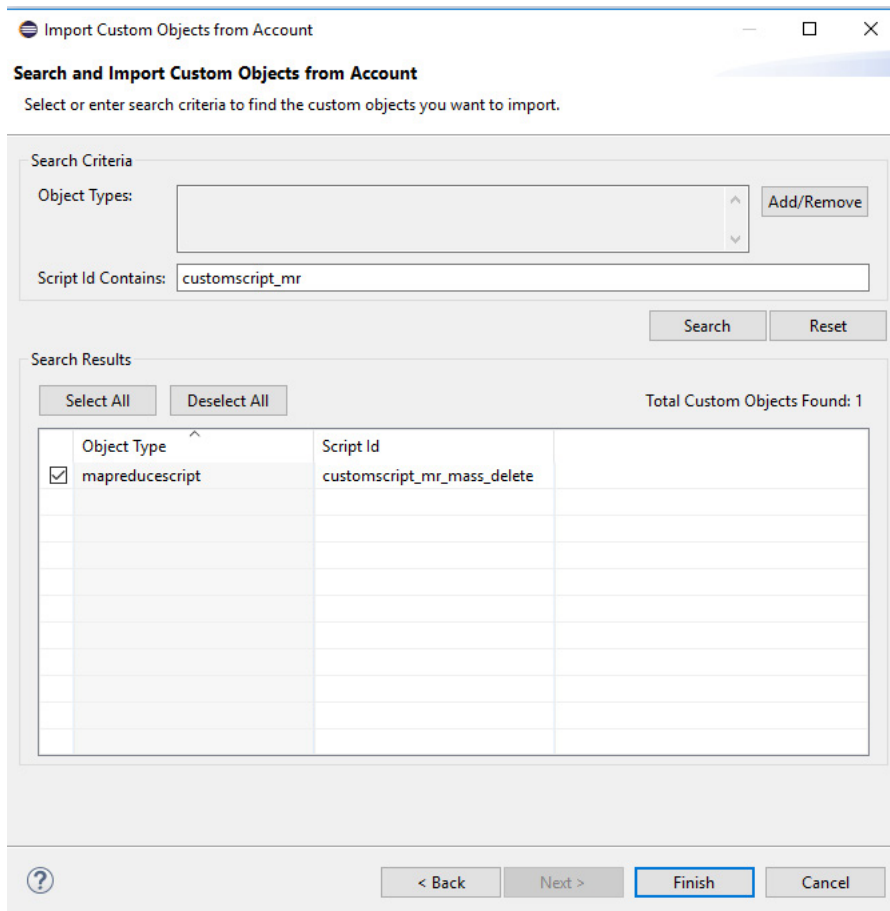
Select All **Deselect All** **Total Custom Objects Found: 1**

Object Type	Script Id
<input checked="" type="checkbox"/> savedsearch	customsearch_mr_delete_sear...

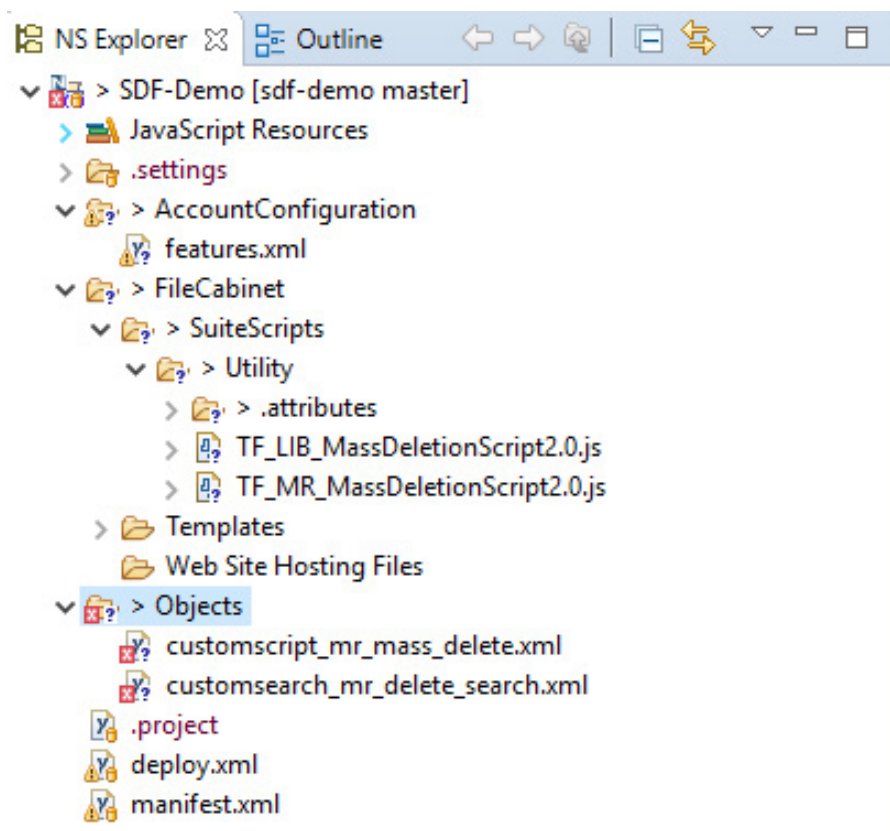
< Back **Next >** **Finish** **Cancel**

- Now we need to import the script object by right clicking Objects folder and go to NetSuite > Import Custom Objects from Account again

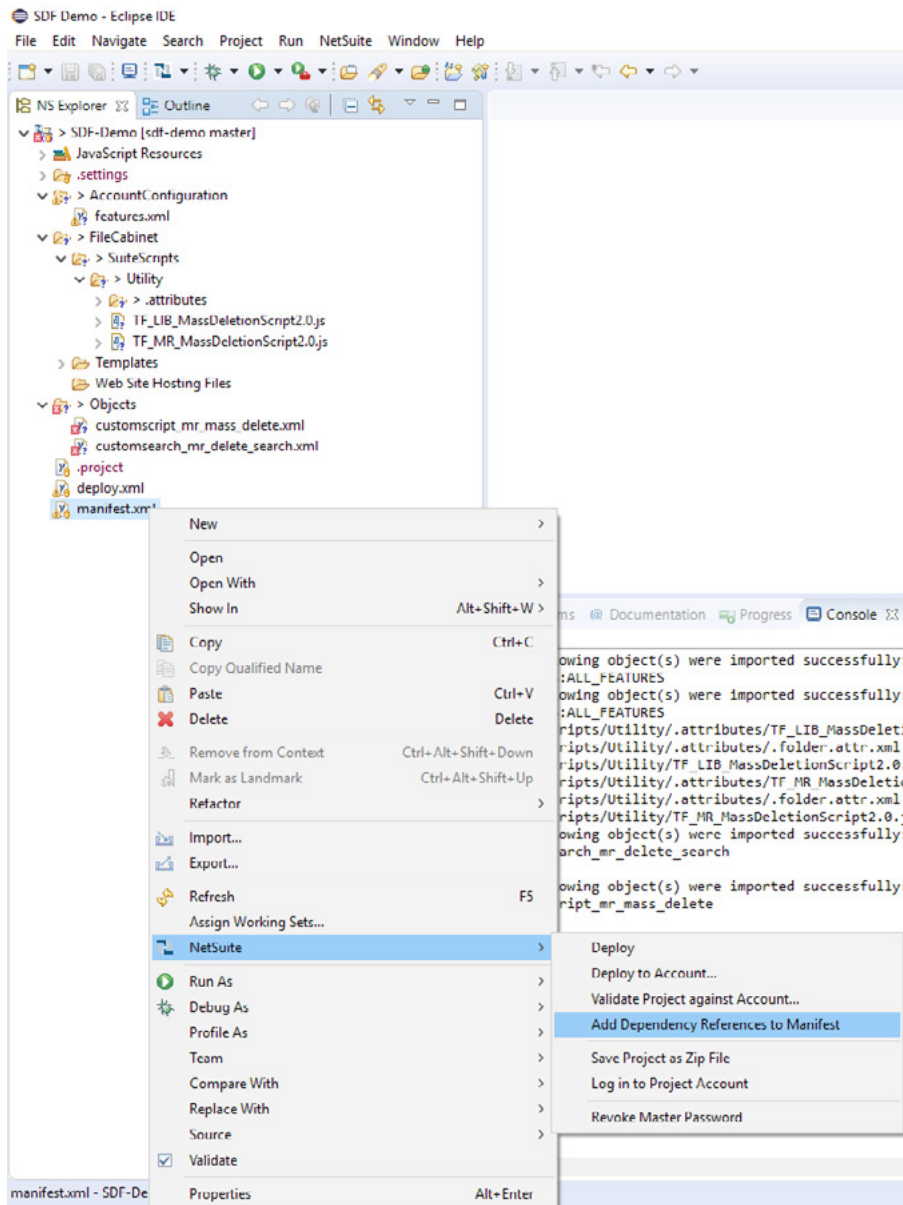
8. Enter and Select the correct object in the window and click finish.



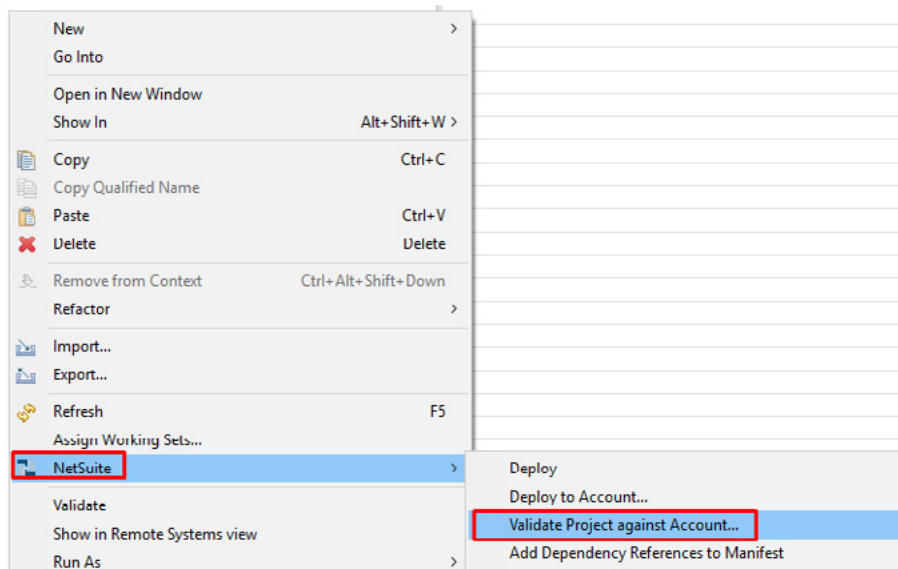
9. Now everything is imported. Your NS Explorer should look like the below image.



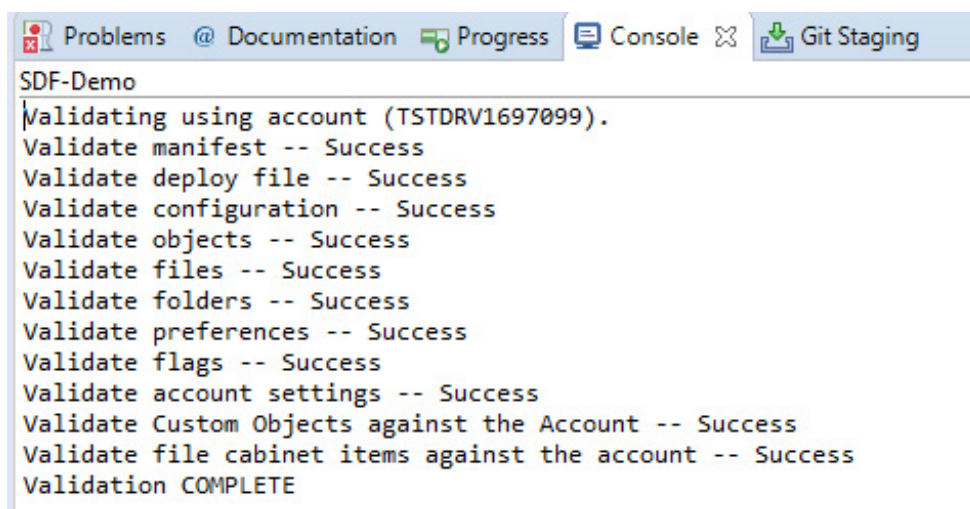
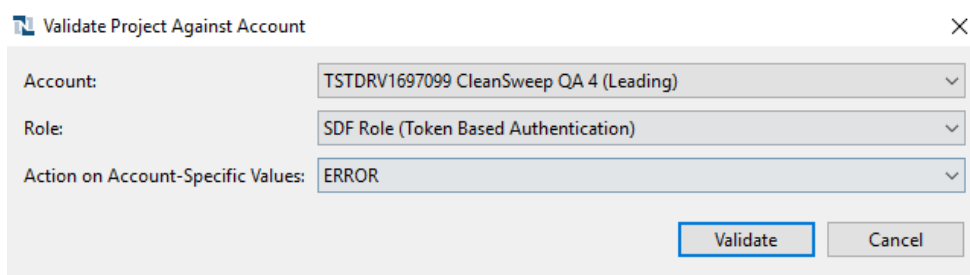
10. Last step for import, we need to update the dependency list. To do this, right click on the manifest.xml file and go to NetSuite > Add Dependency References to Manifest. This will automatically update your manifest.xml to eliminate some of the dependency issue. (In this example, there are not really any dependencies. In a different scenario, you might have to manually update manifest.xml.)



11. Before we deploy the customization to targeted QA account, we want to make sure the deployment would be successful and is not going to cause any errors. To do this, right click on the SDF-Demo project and go to NetSuite > Validate Project against Account



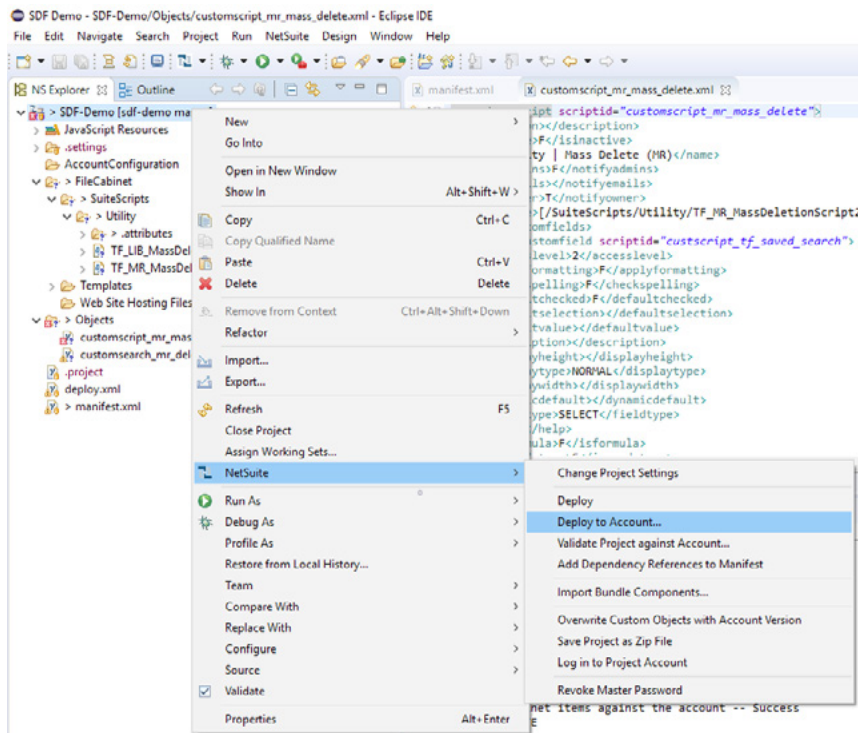
12. In the pop up dialog, select QA account and SDF Role. Since we will deploy this customization to QA, we need to ensure it passes validation. Again, you should check the validation message in console.



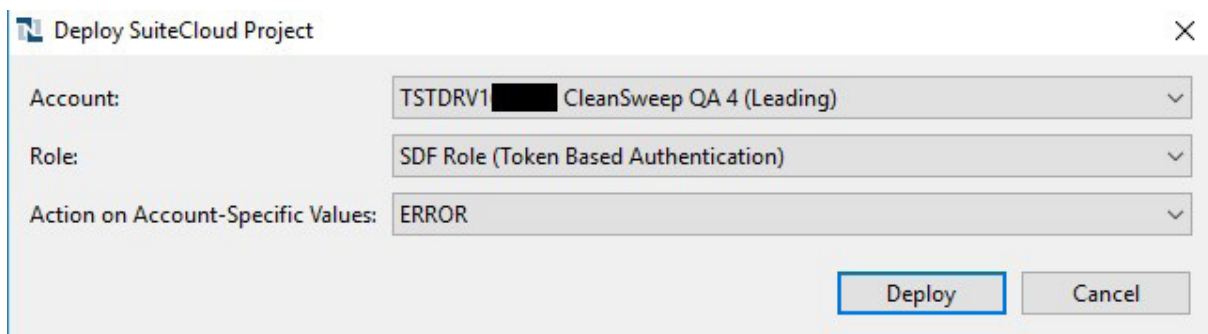
Deploy to Target Account

We are finally at the point to deploy this change to our QA account now that the validation is complete.

1. Right click on SDF-Demo project and go to NetSuite > Deploy to Account



2. Change Account to QA and Role to SDF Role and Click Deploy.



3. Validate customization has been successfully deployed to target account from console logs.

```

SDF-Demo
2019-05-05 23:59:48 (PST) Installation started
Info -- Account [(PRODUCTION) CleanSweep QA 4 (Leading)]
Info -- Account Customization Project [SDF-Demo]
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
Create object -- customscript_mr_mass_delete (mapredudescrpt)
Create object -- customsearch_mr_delete_search (savedsearch)
Create object -- customscript_mr_mass_delete.custscript_tf_saved_search (scriptcustomfield)
Create object -- customscript_mr_mass_delete.customdeploy_mr_mass_delete (scriptdeployment)
2019-05-05 23:59:55 (PST) Installation COMPLETE (0 minutes 7 seconds)

```

4. Log into NetSuite and verify all changes has been deployed

ORACLE NETSUITE

Search

Activities Box Files Payments Transactions Lists Reports Analytics Customization Documents Setup Support Demo Framework Fixed Assets SuiteSocial Sales Knowledge Base

Scripts View Deployments

[New Script](#)

FILTERS

TYPE: MapReduce API VERSION: All SCRIPT FILE: TF_MR_MassDeleteScript2.0.js FROM BUNDLE:

SHOW INACTIVES							TOTAL
EDIT VIEW	NAME	FROM BUNDLE	ID	API VERSION	DEPLOYMENTS	SCRIPT	LIBRARY SCRIPT
Edit View	Utility Mass Delete (MR)		customscript_mr_mass_delete	2.0	Deployments	TF_MR_MassDeleteScript2.0.js	

Script Deployment

[Edit](#) [Back](#) [Actions](#)

SCRIPT
Utility | Mass Delete (MR)

TITLE
Utility | Mass Delete (MR)

ID
customdeploy_mr_mass_delete

☒ DEPLOYED

STATUS
Not Scheduled

SEE INSTANCES
[Status Page](#)

LOG LEVEL
Debug

EXECUTE AS ROLE
Administrator

PRIORITY
Standard

CONCURRENCY LIMIT
2

☐ SUMMIT ALL STAGES AT ONCE

YIELD AFTER MINUTES
10

DIFFER SIZE
1

[Schedule](#) [Parameters](#) [Execution Log](#) [System Notes](#)

SAVED SEARCH FOR DELETE
Mass Delete Search

TITLE	FROM BUNDLE	ID	TYPE	OWNER	ACCESS
Mass Delete Search		customsearch_mr_delete_search	Transaction	RonHong Zhu	Public

5. Don't forget Commit and push all the changes to Git

That concludes this example for deploying Mass Delete Customization from our Dev account to QA account. We hope that you found this example helpful.

Additional Notes for This Example

1. Some of the steps only apply to Account Customization. The process for SuiteApp will be slightly different for the initial project setup, and you will need to supply your Publisher ID.
2. Import AccountConfiguration may not be necessary if the settings are the same between accounts. Using SDF role may cause error due to permission issues. If that's the case, switch to Administrator role for AccountConfiguration for import and export.
3. Validate against Account action is more for dependency check. It does not guarantee deployment will be successful.
4. Always log into NetSuite and check deployment result after SDF deployment.
5. This demo does not follow git best practices and uses only the master branch for demo simplicity
6. There are other ways to import/export project to Git. This is just the writer's way of setting things up.
7. We recommend committing the *.project* since the recent release of Eclipse requires this file for a project. If this particular file is missing, you will run into errors while initializing the SuiteCloud project in Eclipse.
8. To create a new project out of existing Git repository, you can first clone the repository from Git perspective. Then go to File > Open Projects from File System and select the correct root. If the correct root is selected, you will see a SuiteCloud project added to your Explorer. (You must have the *.project* file in the root folder.)

SDF Gotchas & Our Experience

NetSuite has worked to address many of the issues that we discovered on different projects. They continue to improve upon SDF and we are thankful that many of these challenges have been addressed. Below are some of the remaining issues that you may experience :

- **Workflows**

- Permissions - Importing Workflows from an environment or deploying to a target environment requires permissions for your deployment role for all transaction types that are involved in the workflow.
- Workflow Action Script return types - Issue deploying workflows with workflow actions with "List/Record" return types when the value is a List. Netsuite expects a record type and rejects the list. The workaround for this is to remove any return type, deploy, then manually reconfigure the return type.
- Account Specific Values - Be aware that Account Specific Values could result in unintended behavior for workflows. For example, in an Expense Approval workflow, if the chart of accounts does not match between environments, it may use a seemingly random account for approved expenses.

- **Roles** - All roles centers are not supported and not all sublists are supported. Most notably, employee and custom center are not supported and the following sublists are not supported:

- Subsidiaries field
- Forms sublist
- Searches sublist
- Dashboard sublist

NOT RECOMMENDED - The workaround for role center types that are incompatible is to create the role in the target environment with the same ID and make sure the role center type is the intended type. Next, use the console to set the role center type to "BASIC", then save the role. Edit the XML of the imported role and set the role type in the XML to "BASIC". After deployment, set the role center type back to the intended value. We still encountered difficulties with this process, but it was the most consistent solution available.

- **Deployment Roles** - When deploying with SDF the user deploying will need a role with permissions to various records and objects and their dependencies as well as the ability to authenticate and make web service calls to NetSuite preferably using Token Based Authentication. When deploying this creates a Chicken and Egg paradox as you can't deploy the role needed to deploy to the account without deploying. A native or pre-existing role must be created for an SDF deployment to solve this issue. Previously the Administrator role could be used for convenience or as a last resort but with Two Factor Authentication mandatory for any highly privileged roles now this disallows the role from being used. There are a few options and none are very streamlined

1. Manually create a role in the target account. While the most tedious option it is also the most straightforward
2. Install an SDF developer role from a suitebundle. There are many pre-existing bundles for an SDF friendly role. Installing one of these or creating your own SDF developer role bundle can save you a lot of time and headache in the future
3. Create a custom role based on the Native Developer role. The native developer role contains highly privileged permissions and has the same issues as the Administrator role. However you can strip out a few permissions and create a custom role pretty easily that doesn't have mandatory two-factor authentication

- Access Token Management
- Integration Application

This role will likely need to be altered with additional permissions but works for simple projects and as a good starting point

- **Error Messages** - When an SDF operation fails, it can and will be vague, confusing, unintuitive, or a combination of all three. Focus on resolving any and all small issues with the project being deployed and work through each issue.
- **Ensure Uniqueness of script IDs** - One issue encountered was a nebulous error message while trying to deploy a script - "Invalid recordtype reference key CUSTOMER". However, we found that the underlying issue was that there was a script deployment with the ID "customdeploy1" that would not be unique in the target environment, resolving the issue. In general, validate that the IDs used for your customizations are not default IDs, such as "custbody1".
- **Forms** - The difficulty of migrating forms scales with the level of customization in the source account. As they are currently implemented, an SDF Form XML contains a reference to every customization available to be associated with the form in the account, regardless of whether the form uses the field or not. In more complex accounts, this can lead to an extremely large and dependency heavy XML.

- **Saved Searches** - Saved Searches deployed without a "Publish Search" permission defined in the deployment role will be automatically switched from "Public" to "Private" during deployment. This can break a lot of functionality so it is highly recommended that this permission always be included for a deployment role.

Validate Against Account: As we mentioned, you should always use Validate against Account before deployment. However, this feature does not guarantee successful deployment even if it returned a positive result since it only checks against dependencies within project. For example, if there is an issue with permissions, the feature will be unlikely to return an error. We hope that NetSuite could improve the Validate against Account feature so that it can give users warnings as if the change/customization were deployed - especially considering this is the last line of defense before actual deployment.

Closing Remarks

Be sure to check back from time-to-time as we continue to update this guide with new content and information. NetSuite is continuing to build upon this SuiteCloud Development Framework and adding new exciting features you will want to take advantage of. As of this writing, NetSuite is continuing to add more capabilities to round out some of the gaps we have discovered and pain-points we have raised. We at Techfino eagerly look forward to the day where all changes in an account can be managed in version control, deployed via SDF, and automatically validated unit tests and end-to-end integration and functional test cases ensuring quality from end-to-end.

If you are considering using SDF and would like our assistance, we would be more than happy to assist you. Or, if you have any questions, corrections or comments, please send us a note at contact@techfino.com We love hearing from NetSuite customers, practitioners and partners alike!



Who is Techfino?

Techfino LLC is a passionate group of techno-functional NetSuite experts with an innate desire to help companies of all sizes scale and grow. Headquartered in Philadelphia, this national consultancy specializes in designing, implementing, and managing best-in-class cloud solutions. From NetSuite licensing, to training, to support, to integrations and optimizations, Techfino is hyper-focused on solving the demanding challenges of ever-changing business models, bringing specialized expertise in retail, wholesale, manufacturing, professional services, non-profit, commerce and beyond. Techfino is also proud to have developed several proprietary data management products for NetSuite including Cleansweep File Manager and Archive tools.

For more information about Techfino, their products and service offerings, visit www.techfino.com

"Each of our employees has an innate curiosity to dissect challenges and discover their solutions. That's what keeps our team up at night and keeps us innovating every single day."



Bryan Willman
Managing Partner

www.techfino.com
contact@techfino.com

(877) 563-1405



www.linkedin.com/company/techfino