Distributing Integration Records In SDF

SDN Solutions Engineering Team

November 2021

Contents

Introduction

This document demonstrates how to create a SDF SuiteApp that includes an Integration Record, and ship it using the SuiteApp Control Center.

The Integration Record (IR) is a native NetSuite record that provides many functionalities and benefits to NetSuite customers when using Integrated and Hybrid SuiteApps. The IR is critical to these types of SuiteApps that use SuiteTalk APIs (SOAP, REST, RESTlet) due to its key features of facilitating authentication methods and execution logs.

The SDF support for the IR has been available since NetSuite version 2020.2, thus allowing integrated SuiteApps and Hybrid SuiteApps to enjoy the same convenience and user experience as native SuiteApps during the installation phase and subsequent upgrade phases. Starting in the 2022.1 cycle of Built for NetSuite (BFN), each Integrated SuiteApp and Hybrid SuiteApp must ship its dedicated IR in either a bundle or a SDF SuiteApp (preferred). **Creating an IR on each customer's production accounts without using a bundle or SDF SuiteApp will not be BFN compliant.**

This new BFN mandate does not apply to native SuiteApps that do not use any SuiteTalk APIs, neither does it apply to SuiteApps that use SuiteAnalytics Connect.

Disclaimer

The step-by-step procedures indicated and screenshots in this document are taken from NetSuite release 2021.1. The recommendations in this document are from the best practices gathered by the SDN Solutions Engineering team. The processes, screenshots, and link descriptions may vary slightly from the latest versions of NetSuite and SuiteApp Control Center.

Getting your NetSuite Account Ready

Before an ISV can create and use an Integration Record, the account administrator must ensure that the following are enabled and set up accordingly in your NetSuite account. Furthermore, ensure that your Publisher ID is associated with your developer/publisher account otherwise you will not be able to create new or modify existing SuiteApps.

SuiteCloud Features to Enable

The following SuiteCloud features must be enabled in your development/publisher and QA SDN accounts. These features are required for SDF to work and distribute your Integration Record.

To enable the features, go to **Setup → Company → Enable Features** and click on the **SuiteCloud** subtab. Check the features below and click on the **Save** button.

- Token-Based Authentication
- SuiteCloud Development Framework
- SuiteApp Control Center

Prerequisite Roles and Permissions

Assign **User Access Tokens** permission to your existing role or create a new role with the permission. It is preferred to assign the **Developer** role to users who will develop and test using SDF. The Developer role has an existing set of permissions for development purposes including User Access Tokens. If the default Developer role does not suffice, it can be customized as necessary.

For deployment and distribution purposes, the best practice is to create a **Custom Role** with the least privileges and ship it with your SDF SuiteApp. This custom role must be used by the customer when creating the Access Tokens. You must test the custom role in your QA account to ensure that Access Tokens are created and integration is successful.

NetSuite SuiteCloud SDK

For the purposes of this tutorial, you must have SuiteCloud Extension for Visual Studio Code by Oracle Corporation, WebStorm with NetSuite SuiteCloud IDE plugin (2021.1 or later), or Node.js installed to create an SDF SuiteApp project, import the Integration Record from the NetSuite UI, and deploy them to your TSTDRV account. For instructions about installing the SuiteCloud Extension to Visual Studio Code, SuiteCloud SDK plugin to WebStorm, or installing Node.js, please refer to the SDF Tutorials for Visual Studio Code, WebStorm, and Node.js documents which can be downloaded from the Technical Onboarding Site (https://sites.oracle.com/site/SDN_Site/).

Integration Record Installation Mechanism

When an integration record (IR) is created in the Development account by a developer filling out the Integration form, two things happen:

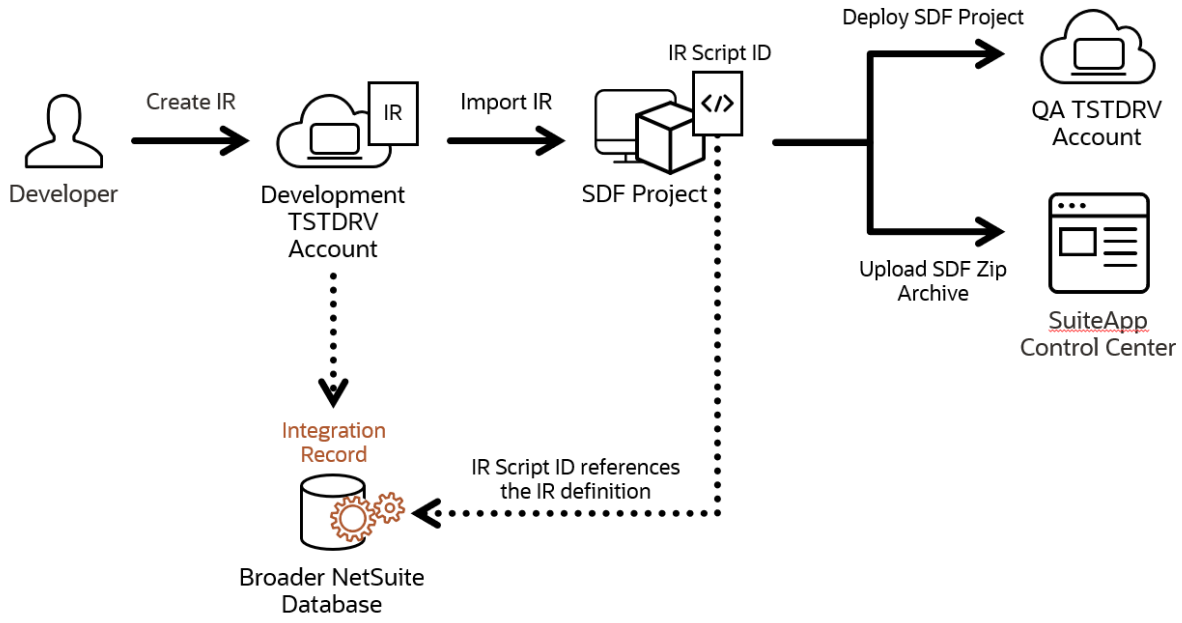
- An IR is added to the Development account.
- A definition of that IR is created within the broader NetSuite database.

After the development phase has completed, the IR needs to be imported into the SDF Project from the Development account. When importing the IR, the xml file will only have the *scriptid* which is the reference to the IR (see below).

```
<integration scriptid="custinteg_1a2b3c4d5e6f">
</integration>
```

When deploying the SDF Project or installing the SDF SuiteApp that contains the IR, NetSuite references the IR definition from the broader NetSuite database and creates the IR in the target NetSuite account.

The following illustrates the IR mechanism from development to deployment/installation stage:



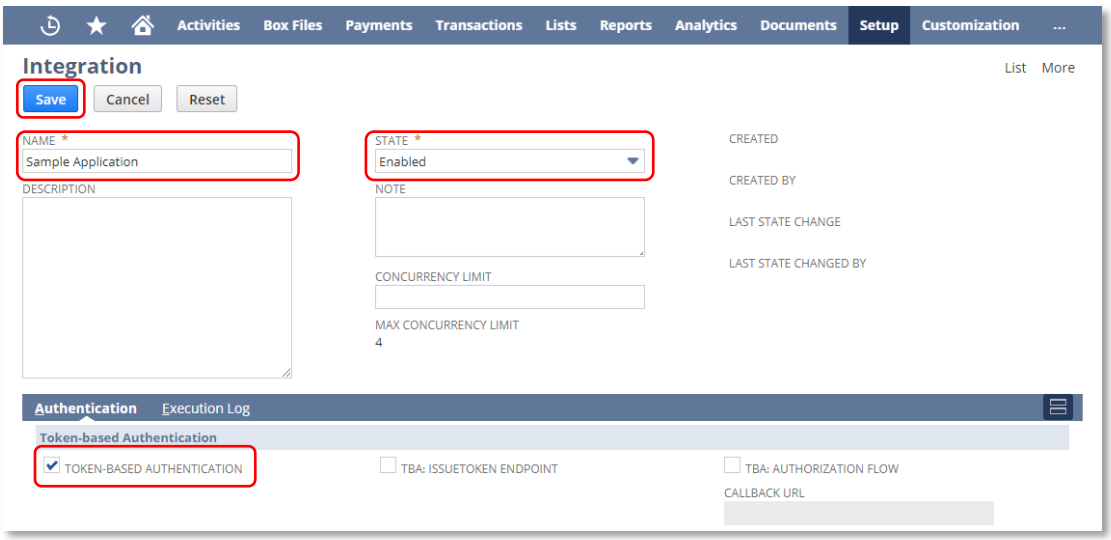
Creating an Integration Record

The first step in integrating your SuiteApp to NetSuite is by creating the Integration Record (IR). As of NetSuite version 2021.2, you can only create an integration record from the NetSuite UI.

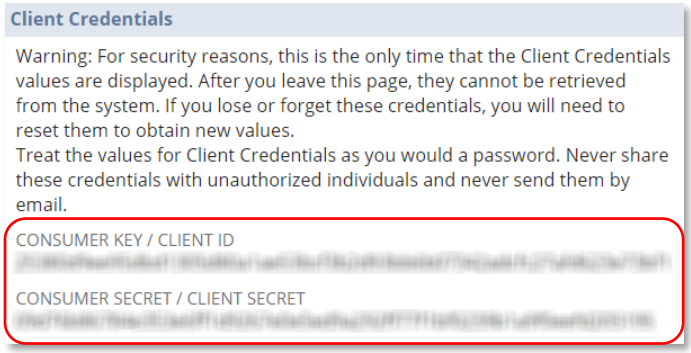
To create an IR, follow these instructions:

- In the NetSuite UI, go to **Setup → Integration → Manage Integrations → New**.

- In the **Integration Page,** input the desired information for your SDF SuiteApp. The **Token-Based Authentication** option under *Authentication* subtab must be ticked. Click on the **Save** button.



- Click on the **Save** button. After saving the IR, NetSuite generates a *Consumer Secret* and *Consumer Key* for the integration.



IMPORTANT:

- *Since the client credentials are showed only once and cannot be retrieved, you need to copy those values to a secure location as you will need them later. It is important that the consumer secret and key are not shared to unauthorized individuals.*
- *You must only use one integration record per SuiteApp Application ID. Refer to the topic **SDF SuiteApp Project Structure** on any of the SDF tutorials to learn more about SuiteApp Application IDs which can be downloaded from the Technical Onboarding Site. You can also refer to SuiteAnswers for more information about SuiteApp Application ID.*

NOTE: *If you want to test the integration during the development phase, an Access Token should be created for the external application to gain access to the NetSuite account. Refer to the topic **Deploying the Integration Record** for instructions on creating the Access Token for the IR.*

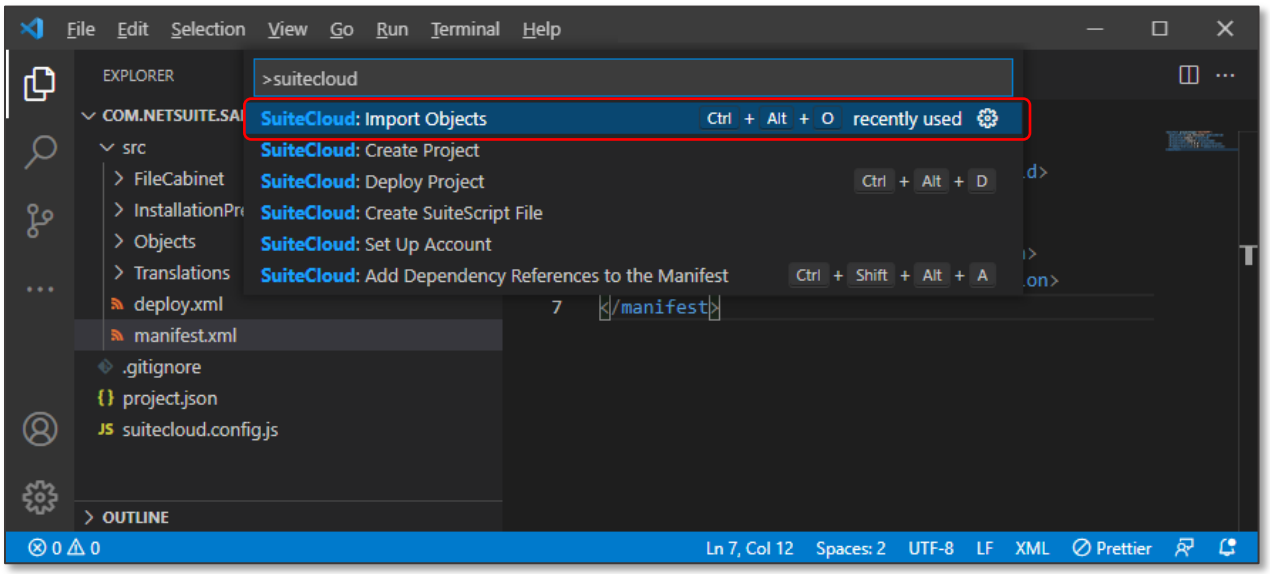
Importing the Integration Record

Once the development phase has completed, the SDF SuiteApp Project needs to be tested in your SDN trailing QA account. It is recommended that you only import the Integration Record (IR) from the Development account to the SDF SuiteApp Project when you are ready to perform the QA testing. You cannot deploy the SDF SuiteApp Project with the IR to the same Development account since the object already exists. The SDF SuiteApp Project has to be redeployed without the IR.

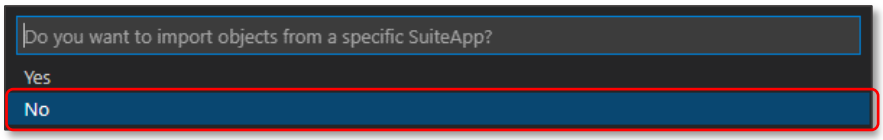
To import the IR, follow these instructions:

Visual Studio Code IDE

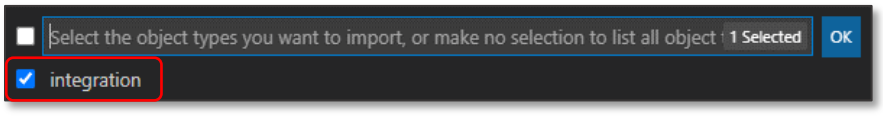
- In Visual Studio Code, select a file from your SuiteApp project to make it the active project.
- Click on **View → Command Palette**.
- In the **Command Palette** field, please perform the following:
  - Type ***SuiteCloud*** and select ***SuiteCloud: Import Objects***.



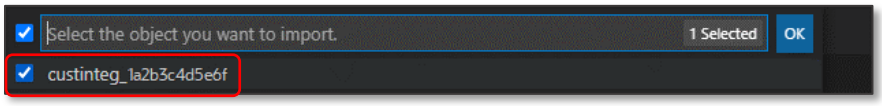
  - Select ***No*** when prompted to import objects from a specific SuiteApp.



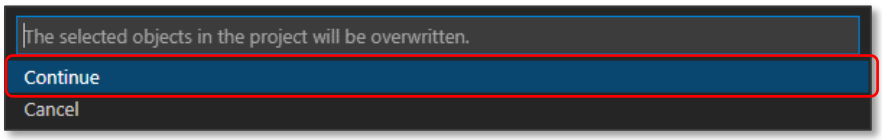
  - Type ***integration*** and tick the checkbox.



  - Enter the full or partial script ID of your integration record object.
  - Locate and tick the checkbox of the desired integration record object.



- Select ***Continue*** to import the integration record object from your Development trailing account.

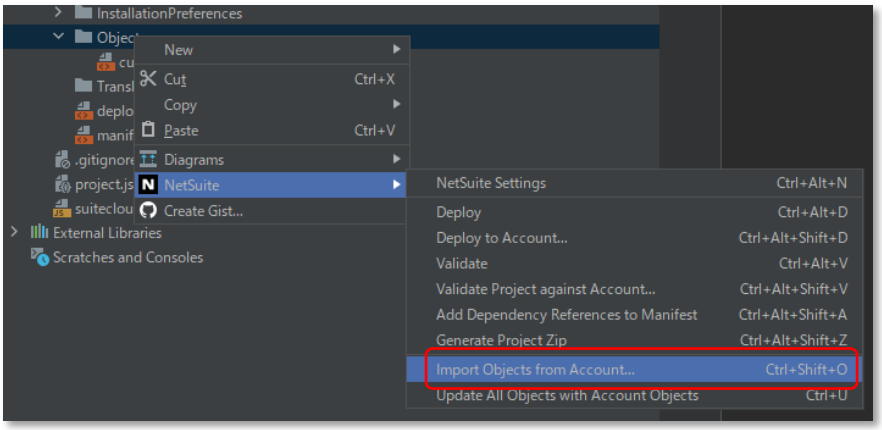


The import progress log is displayed in the **Output** tab.

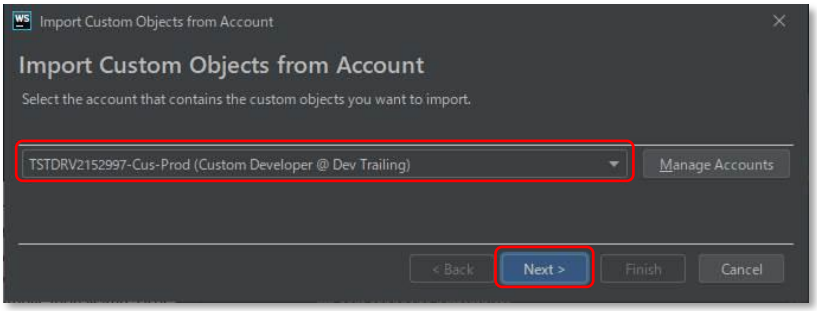
```
The imported objects will overwrite the project objects
The following objects have been imported:
    - integration:custinteg_1a2b3c4d5e6f
```

WebStorm IDE

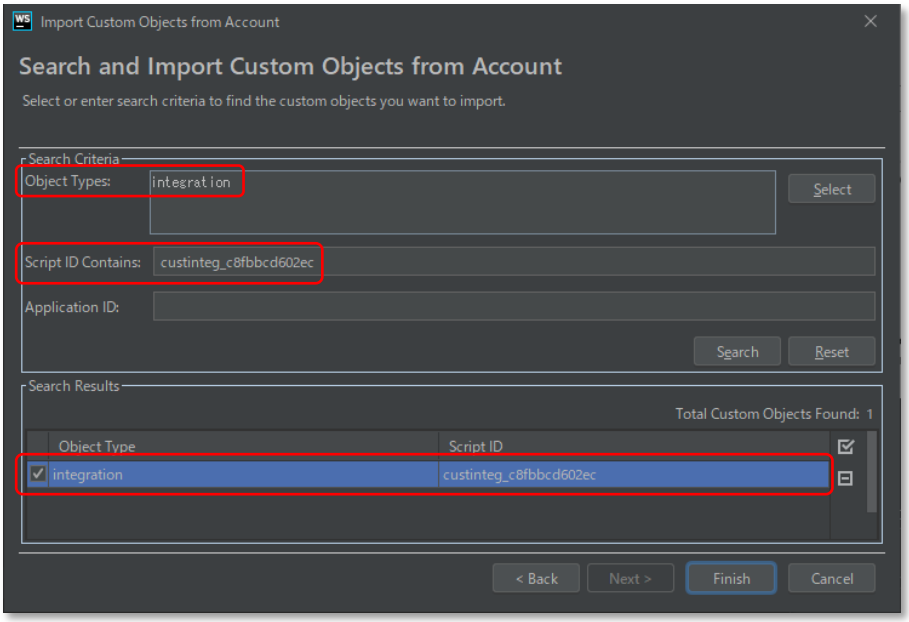
- In WebStorm, right-click on the SDF SuiteApp Project and go to **Context Menu → NetSuite → Import Objects from Account**.



- From the drop-down list, select your Development trailing account to import the IR from. Click on the **Next** button.



- Select the IR object you want to import. You can specify a search criteria to trim down the listed objects by Object Type, Script ID, and/or Application ID.



- In the Search Results, tick the checkboxes to select the IR you want to import. Click on the **Finish** button when done.

CLI for Node.js

- Launch the Command Prompt or terminal.
- Navigate to your desired project. Ex. *cd\cliworkspace\com.netsuite.sampleapp*
- Type and run ***suitecloud object:import -i***

```
$ suitecloud object:import -i

? Do you want to import objects from a specific SuiteApp? No
? Do you want to import all object types? No
? Select the object types you want to import. Integration
? Do you want to enter a script ID to filter your list? Yes
? Enter the full or partial script ID. custinteg_1a2b
? Select the objects you want to import integration:custinteg_1a2b
3c4d5e6f
? Select the folder where you want to import the objects. /Objects
? All the selected objects in the project will be overwritten with the
objects from the account. Do you want to continue
? Yes
The following objects have been imported:
    - integration:custinteg_1a2b3c4d5e6f
```

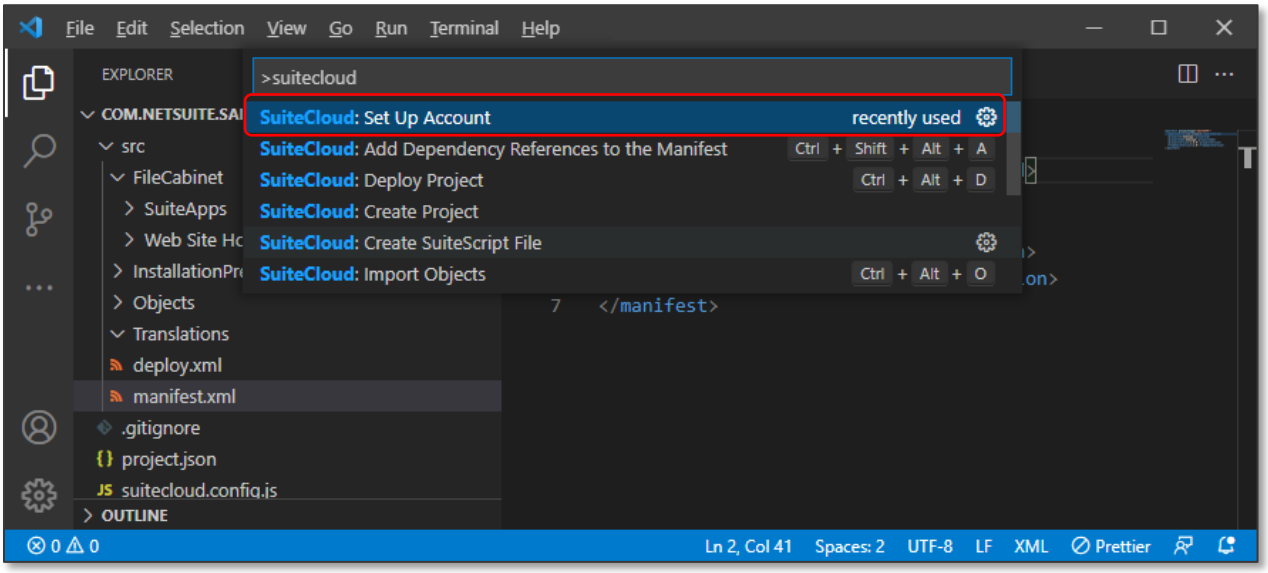
Testing the Integration Record

The next step is to deploy the Integration Record (IR) to the QA trailing account for testing. It is important to test that the IR works as expected and that your external application can access NetSuite.

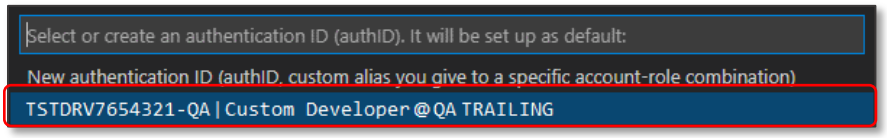
To deploy the IR, follow these instructions:

Visual Studio Code IDE

- In Visual Studio Code, select a file from your SuiteApp project to make it the active project.
- Click on **View → Command Palette**.
- In the **Command Palette** field, type ***SuiteCloud*** and select ***SuiteCloud: Setup Account*** to link the SDF SuiteApp project to your QA account.



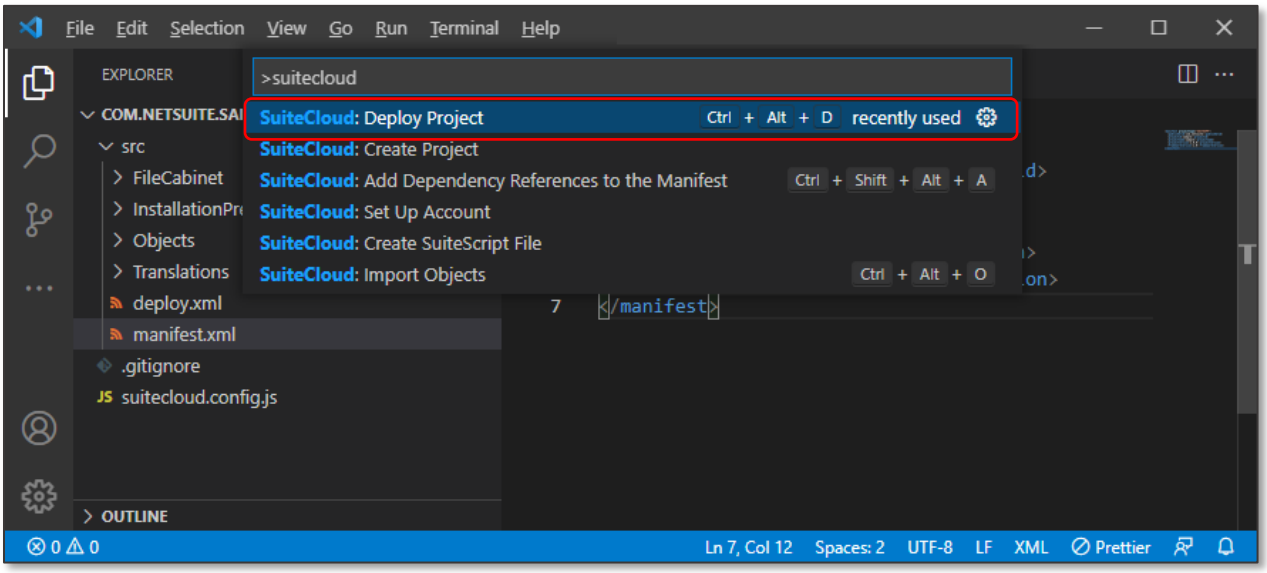
- Select the account-role combination in your QA trailing account from the dropdown list.



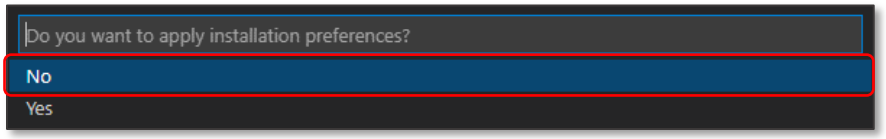
The account setup progress log is displayed in the **Output** tab.

```
The default account for the current project was successfully set to
TSTDRV7654321.
```

- Click on **View → Command Palette**.
- In the **Command Palette** field, enter ***SuiteCloud*** and from the dropdown list, select ***SuiteCloud: Deploy Project.***



- Select ***No*** when prompted to apply installation preferences.

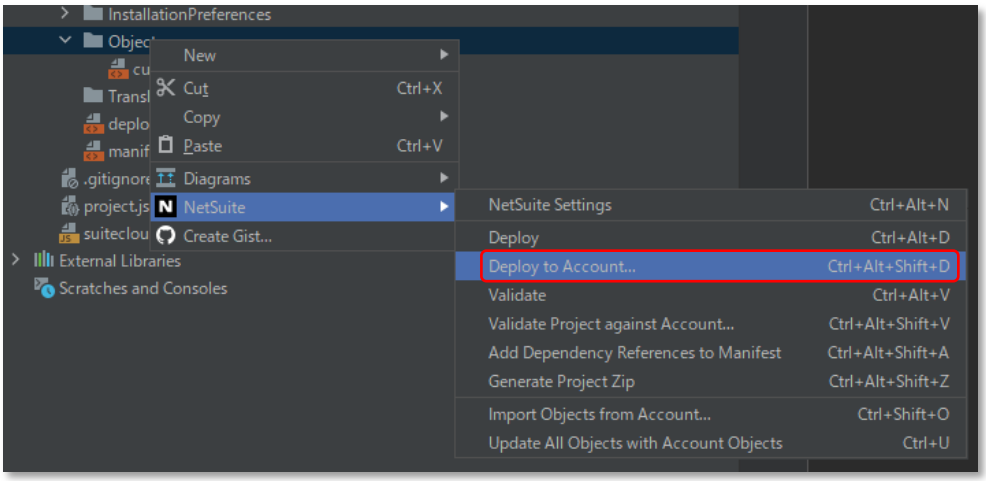


The deployment progress log is displayed in the **Output** tab.

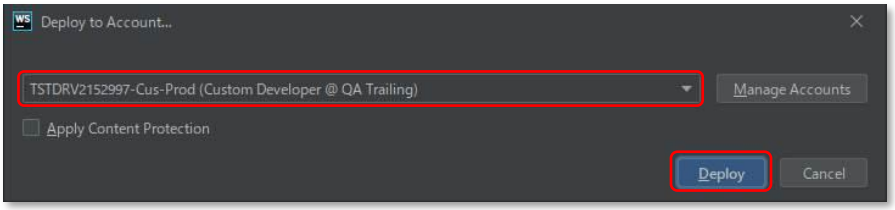
```
The "C:\vscodeprojects\com.netsuite.sample\src" project has been de-
ployed without applying content protection.
The deployment process has finished successfully.


Deploying to TSTDRV7654321 - QA Trailing - Custom Developer.
2021-02-27 23:46:30 (PST) Installation started
Info -- Account [(PRODUCTION) QA Trailing]
Info -- Publisher ID [com.netsuite]
Info -- SuiteApp [com.netsuite.sample (1.0.0)]
Info -- Framework Version [1.0]
Validate manifest -- Success
Validate deploy file -- Success
Validate objects -- Success
Validate files -- Success
Validate folders -- Success
Validate translation imports -- 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
Validate translation imports against the account -- Success
Info -- Current installed version [1.0.0]
Begin deployment -- com.netsuite.sample (1.0.0)
2021-02-27 23:46:32 (PST) Installation COMPLETE (0 minutes 2 seconds)
```

WebStorm IDE

- In WebStorm, go to **Context Menu → NetSuite** and click **Deploy to Account**.



- From the drop-down list, select the QA trailing account where the SDF SuiteApp Project will be deployed. Click on the **Deploy** button.



CLI for Node.js

- Launch the Command Prompt or terminal.
- Navigate to your desired project. Ex. *cd\cliworkspace\com.netsuite.sampleapp*
- Type and run ***suitecloud account:setup -I*** to link the SDF SuiteApp project to your QA account.

```
$ suitecloud account:setup -i

? Select or create an authentication ID (authID, a custom alias you
give to a specific account-role combination):
  ***The authentication ID that you select or create will be set up as
default. TSTDRV7654321 | QA Trailing [Custom Developer]
This project will use the authentication ID "TSTDRV7654321" as default
for the following company and role: QA Trailing [Custom Developer]. If
you want to change your default credentials, run "suitecloud ac-
count:setup" again.
The account has been successfully set up.
```

- Type and run ***suitecloud project:deploy -i***

```
$ suitecloud project:deploy -i

? Do you want to apply content protection? No
? Do you want to validate locally before deploying? No
The "C:\cliworkspace\com.netsuite.sampleapp\src" project has been de-
ployed
without applying content protection.
The deployment process has finished successfully.
Deploying to TSTDRV7654321 – QA Trailing - Custom Developer.
2021-02-27 23:46:30 (PST) Installation started
Info -- Account [(PRODUCTION) QA Trailing]
Info -- Publisher ID [com.netsuite]
Info -- SuiteApp [com.netsuite.sample (1.0.0)]
Info -- Framework Version [1.0]
Validate manifest -- Success
Validate deploy file -- Success
Validate objects -- Success
Validate files -- Success
Validate folders -- Success
Validate translation imports -- 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
Validate translation imports against the account -- Success
Info -- Current installed version [1.0.0]
Begin deployment -- com.netsuite.sampleapp (1.0.0)
2021-02-27 23:46:32 (PST) Installation COMPLETE (0 minutes 2 seconds)
```

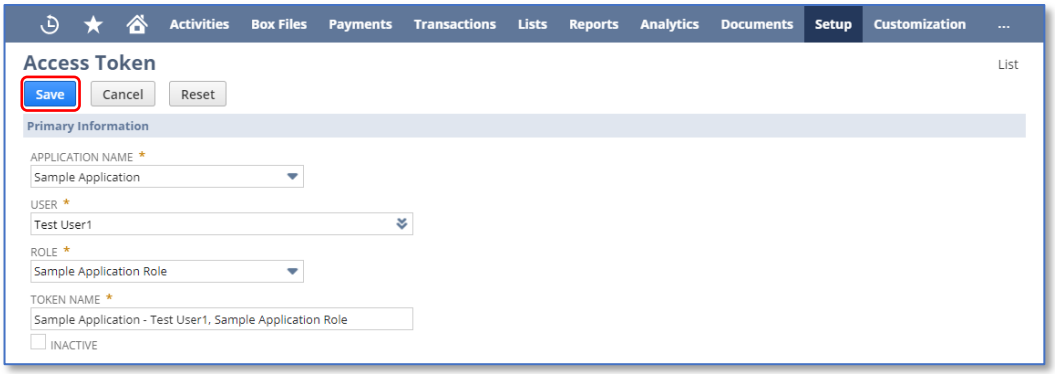
To access the QA trailing account from your external application, you need to create an Access Token for each QA tester.

IMPORTANT: *It is best to assign the final version of the* ***Custom Role*** *with the least privileges that will be shipped with your SDF SuiteApp to each QA tester. This ensures that the custom role works accordingly prior to distribution of the SDF SuiteApp.*

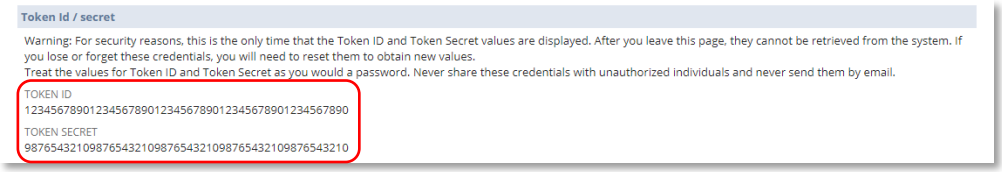
- In the NetSuite UI, go to **Setup → Users/Roles → Access Tokens → New**.

  NOTE: *Users that are assigned with a customized role that has* ***Access Token Management*** *permission can create, assign, and manage a token for other users except tokens for an Administrator role.*

- Select the **Application Name** (the Integration Record), **User**, and **Role**. The *Token Name* is already populated by default (Application Name+User+Role) and can be changed if desired. Click on the **Save** button.



- After saving the Access Token record, NetSuite generates a **Token Secret** and **Token ID** for the access token.



IMPORTANT: *Since the Token ID and Secret are showed only once and cannot be retrieved, you need to copy those values to a secure location as you will need them later. It is important that the Token ID and Secret are not shared to unauthorized individuals. There should only be one access token record per user for each user role.*

Distributing the Integration Record with your SDF SuiteApp

To distribute the Integration Record (IR) with the SDF SuiteApp, you need to create an SuiteApp ZIP archive, define, and upload your SuiteApp in the **SuiteApp Control Center** using the *SuiteApp Release Manager* role. For more information on SuiteApp Control Center and setting up the SuiteApp Release Manager role, please refer to the document ***Managing SDF SuiteApps in SuiteApp Control Center*** which can be downloaded from the APC portal.

To distribute the IR, follow these instructions:

Visual Studio Code IDE

As of 21.2, the SuiteCloud Extension for Visual Studio Code does not have the capability to create a SuiteApp ZIP Archive. Therefore, you need to create it using one of the following methods:

Creating the SuiteApp ZIP Archive using SuiteCloud CLI for Node.js

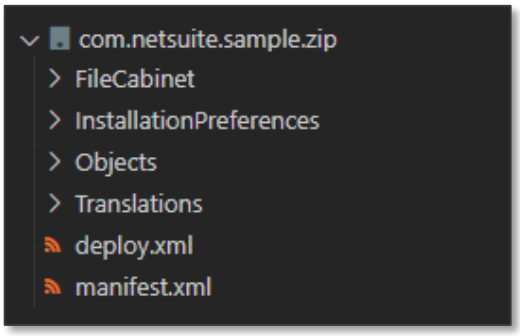
1. Launch the Command Prompt or terminal.
2. Navigate to your desired project. Ex. *cd\vscodeprojects\com.netsuite.sample*
3. Type and run ***suitecloud project:package***.

```
$ suitecloud project:package

The C:\vscodeprojects\com.netsuite.sample \build\com.netsuite.sample-
1.0.0-2021-02-27_00-30-19.zip file has been successfully created.
```

Create the SuiteApp ZIP Archive manually

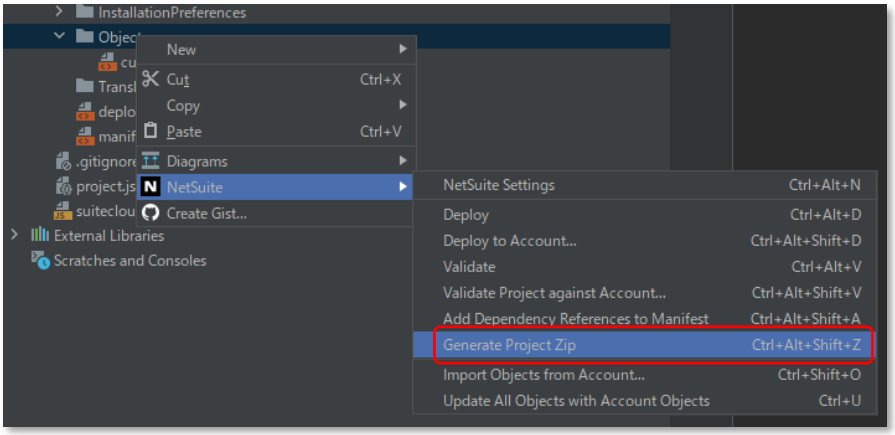
1. Launch the Command Prompt or terminal.
2. Navigate to your desired project. Ex. *cd\vscodeprojects\com.netsuite.sample*
3. Type and run creating zip file command:
   - ***powershell compress-archive src/* [zipped filename]*** if you are using Windows.
   - ***zip -r [zipped filename] src/**** if you are using Linux or MacOS.

When you create a SuiteApp ZIP Archive, make sure that it follows the structure below. The manifest and deploy XML files and other folders should be located under the root folder of the SuiteApp ZIP Archive.



WebStorm IDE

- In WebStorm, right-click on the SDF Project and go to **Context Menu → NetSuite** and click **Generate Project Zip**.



CLI for Node.js

- Launch the Command Prompt or terminal.
- Navigate to your desired project. Ex. *cd\cliworkspace\com.netsuite.sampleapp*
- Type and run ***suitecloud project:package.***

```
$ suitecloud project:package

The C:\cliworkspace\com.netsuite.sampleapp \build\com.netsuite.sam-
pleapp-1.0.0-2021-
02-27_00-30-19.zip file has been successfully created.
```

After generating the SuiteApp ZIP archive, perform the following steps:

- In your Development/Publisher account, log in using the **SuiteApp Release Manager** role.
- In the NetSuite UI, go to **Control Center → My SuiteApps → My SuiteApps** to access the **SuiteApp Control Center**.

  ***NOTE**: For instructions on how to upload your SDF SuiteApp to SuiteApp Control Center, please refer to the document **Managing SDF SuiteApps in SuiteApp Control Center** which can be downloaded from the APC portal.*

References

SDN Technical Onboarding Site

https://sites.oracle.com/site/SDN_Site/

Testing the Integration using POSTMAN

https://netsuite.custhelp.com/app/answers/detail/a_id/86958

SuiteApp Application ID

https://netsuite.custhelp.com/app/answers/detail/a_id/71944#subsect_1522775442