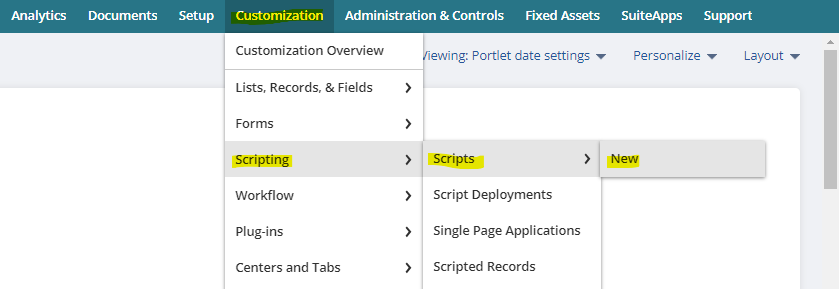
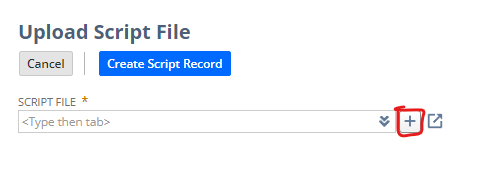
**GETTING STARTED WITH SUITESCRIPT (ASSIGNMENT)**

**1. Create and deploy the user event script using SuiteScript 1.0**

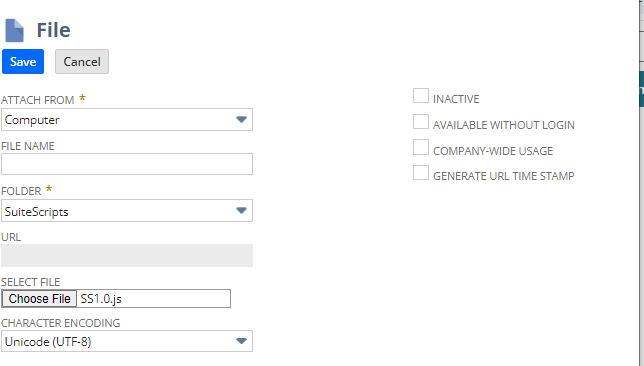
***Step 1*** – Navigate to Customization, Scripting -> Scripts -> New.



***Step 2*** – Click on the '+' icon to add your .js file.

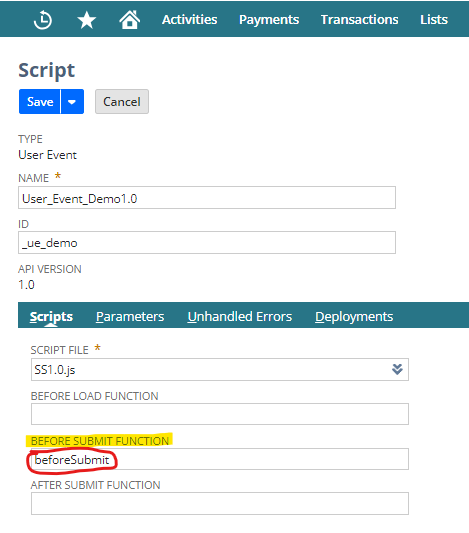


***Step 3*** – Now click on Save.

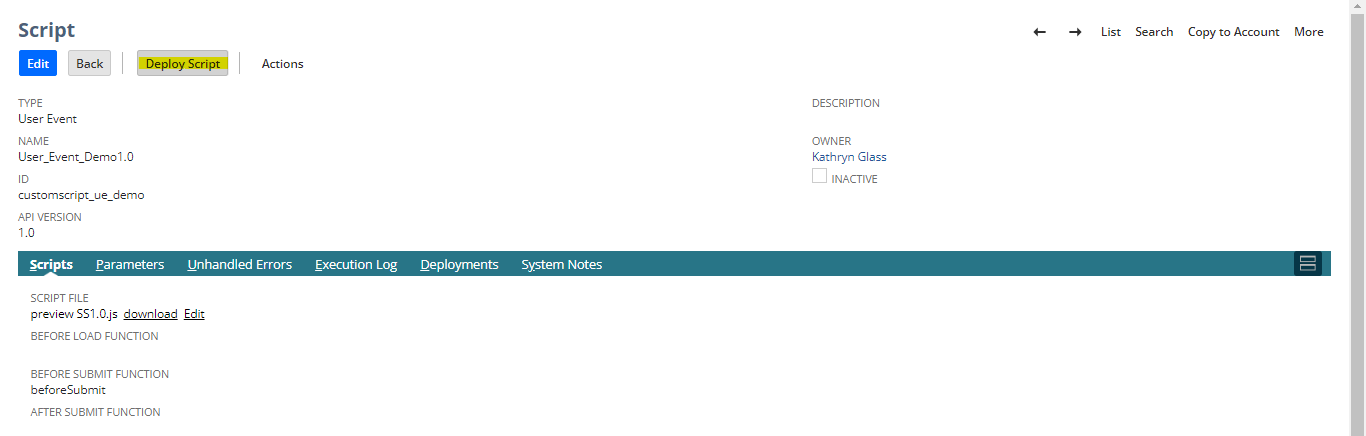


***Step 4*** – Now click on Create Script Record and choose ‘user event script’.

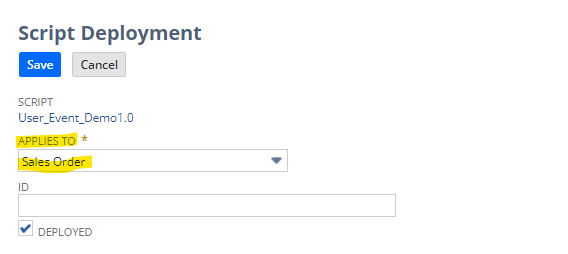
***Step 5*** – Now we will provide the name and ID and then as we want to make this functionality work before submitting, so just write the function name in the before submit function field. Then hit SAVE.



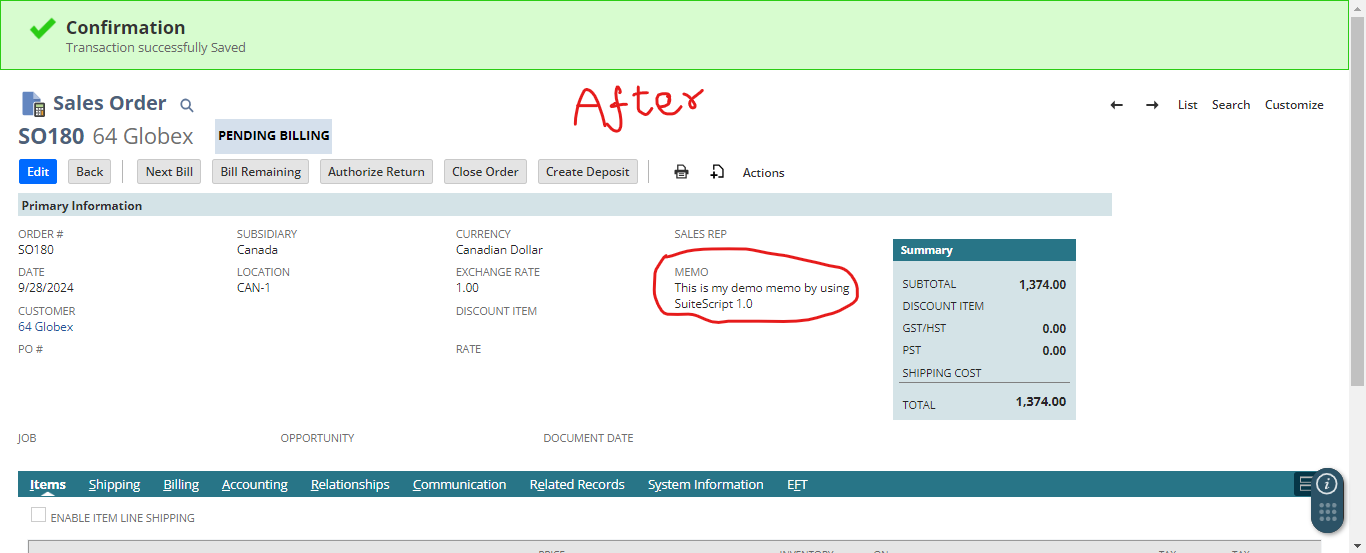
***Step 6*** – Now click on Deploy Script.



***Step 7*** – Now in Script Deployment Select Sales order in applies to section. Now SAVE it.

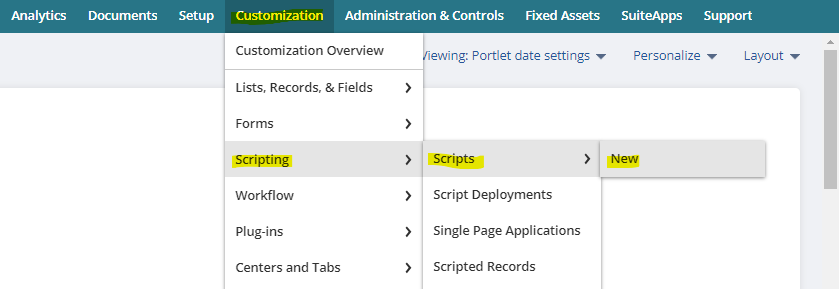


***Step 8*** – Now you can see the resultant output.

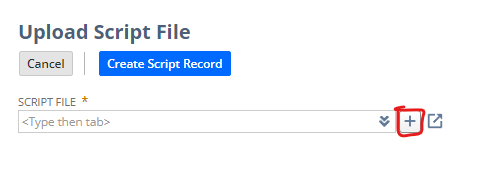


**2. Create and deploy the user event script using SuiteScript 2.0**

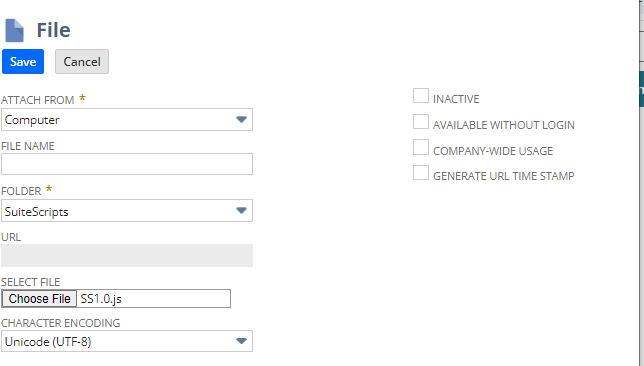
***Step 1*** – Navigate to Customization, Scripting -> Scripts -> New.



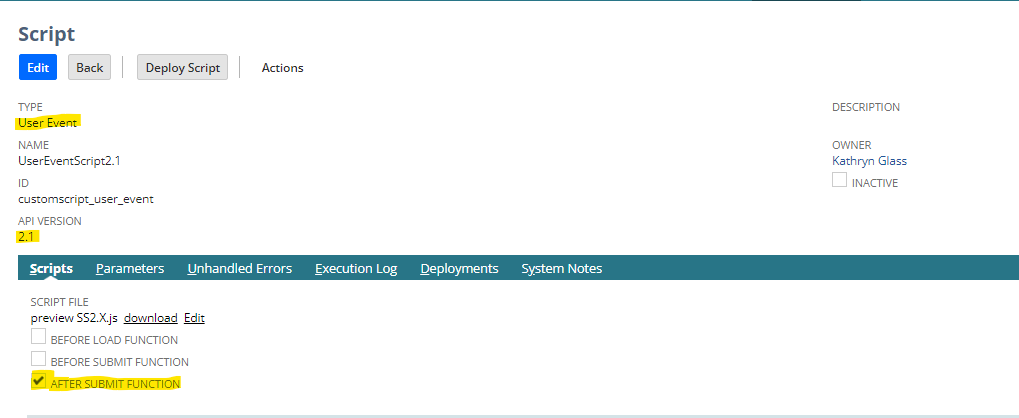
***Step 2*** – Click on the '+' icon to add your .js file.



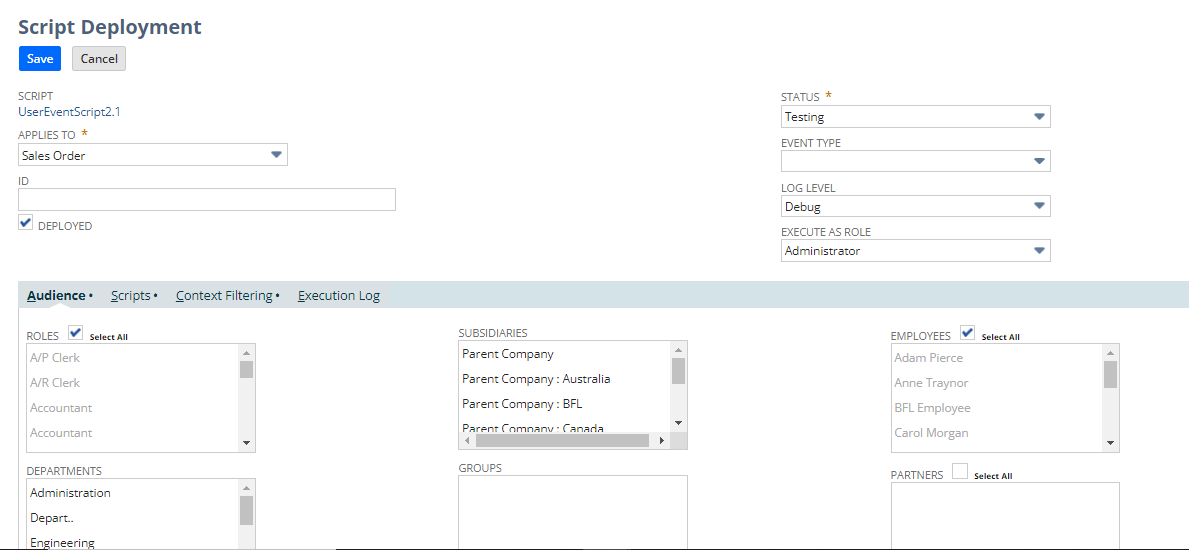
***Step 3*** – Now click on Save.



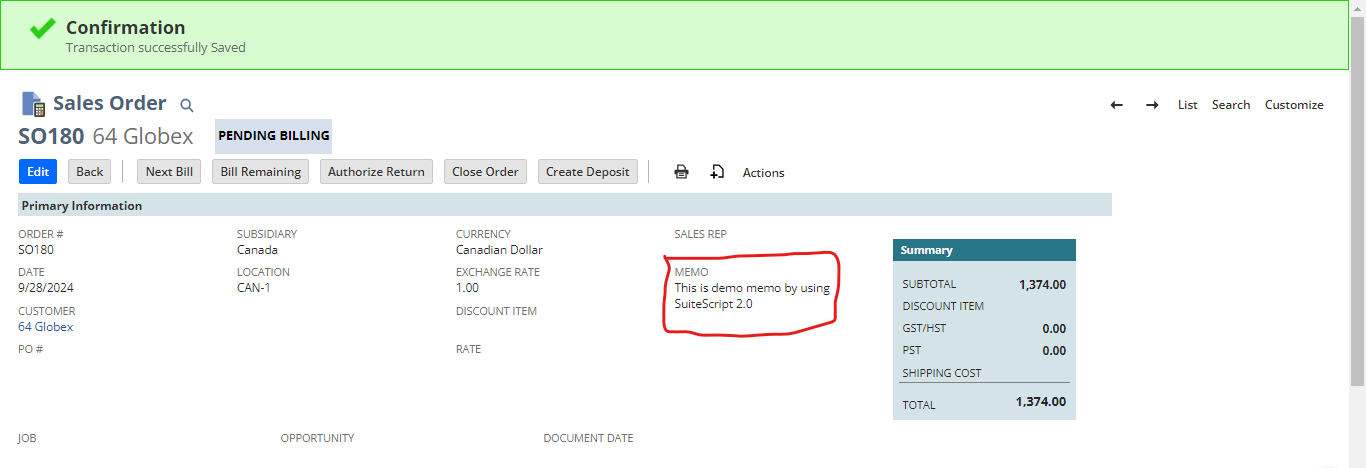
***Step 4*** – Now in SuiteScript 2.0, the type, API version as well as the submission type (After Submission) will be automatically fetched from the .js file we uploaded recently. Now click on ‘Deploy Script’ .



***Step 5*** – Now select the role, subsidiaries, ID, execute as role and click SAVE.



***Step 6*** – Now go to the sales order and click edit and save. Now you will get the desired value in the memo field.



**3. Write API difference of 1.0 and 2.0 version for below modules. Along with this API details (Min 3 API)**

**a. Record**

**b. File**

**c. Search**

**d. Submit Field and lookup field (API)**

**A. Record Module -**

The Record module is used to work with records like customer, invoice, sales order, etc.

***SuiteScript 1.0 API:***

* **nlapiLoadRecord(recordType, recordId):** It loads an existing record.
* **nlapiCreateRecord(recordType)**: It creates a new record.
* **nlapiSubmitRecord(record, doSourcing, ignoreMandatoryFields)**: It saves the record.

***SuiteScript 2.0 API:***

* **record.load(options)**: It loads an existing record.
  + Example: record.load({type: 'customer', id: 1234});
* **record.create(options):** It creates a new record.
  + Example: record.create({type: 'invoice'});
* **record.save(options):** It saves the record after modifications.
  + Example: var id = record.save({ enableSourcing: true, ignoreMandatoryFields: false});

**B. File Module -**

This module allows you to manage and manipulate files in the NetSuite file cabinet.

***SuiteScript 1.0 API:***

* **nlapiLoadFile(fileId)**: Loads a file from the file cabinet.
* **nlapiCreateFile(name, fileType, contents):** Creates a new file.
* **nlapiSubmitFile(file):** Saves the file to the file cabinet.

***SuiteScript 2.0 API:***

* **file.load(options)**: Loads a file by its ID.
  + Example: file.load({id: 1234});
* **file.create(options)**: Creates a new file with a specific name and its type.
  + Example: file.create({name: 'myfile.txt', fileType: file.Type.PLAINTEXT, contents: 'Hello World!'});
* **file.save(options):** Saves the file to the file cabinet.
  + Example: myFile.save();

**C. Search Module -**

The Search module is used to create and run searches to retrieve data from records.

***SuiteScript 1.0 API:***

* **nlapiSearchRecord(recordType, searchId, filters, columns)**: Performs a record search.
* **nlapiLoadSearch(recordType, searchId)**: Loads an existing saved search.
* **nlapiSearchGlobal(searchTerm)**: Performs a global search.

***SuiteScript 2.0 API:***

* **search.create(options)**: Creates a new search with filters and columns.
  + Example: search.create({type: 'customer', filters: [/\* filters \*/], columns: [/\* columns \*/]});
* **search.load(options)**: Loads a saved search by ID.
  + Example: search.load({id: 'customsearch\_my\_saved\_search'});
* **search.global(options)**: Performs a global search.
  + Example: search.global({keywords: 'John Doe'});

**D. Submit Field and Lookup Field API -**

These APIs are used to update or retrieve specific fields on a record without loading the entire record.

***SuiteScript 1.0 API:***

* **nlapiSubmitField(recordType, recordId, fieldName, value)**: It updates a specific field on a record.
* **nlapiLookupField(recordType, recordId, fields)**: Looks up and returns field values from a record.

***SuiteScript 2.0 API:***

* **record.submitFields(options)**: Updates fields on a record.
  + Example: record.submitFields({type: 'customer', id: 1234, values: {email: 'bfl@gmail.com '}});
* **search.lookupFields(options)**: Looks up field values from a record.
  + Example: search.lookupFields({type: 'customer', id: 1234, columns: ['email', 'firstname']});

**4. For invoice, Sales Order, Customer and vendor record get the internal id of:**

**a. 5 Body level fields**

**b. 5 Line level fields**

**c. 3 Sublist IDs**

**A. Invoice Record**

***a. Body-Level Fields:***

1. tranid: Transaction ID (Invoice number)
2. entity: Customer
3. trandate: Date of Invoice
4. memo: Memo/Description
5. duedate: Due Date

***b. Line-Level Fields:***

1. item: Item (for products)
2. quantity: Quantity of the item
3. rate: Rate per unit of the item
4. amount: Total amount for the line
5. taxcode: Tax Code applied to the line item

***c. Sublist IDs:***

1. item: The item sublist where products/services sold are listed.
2. taxdetails: Tax details sublist.
3. apply: The sublist showing payments applied to the invoice.

**B. Sales Order Record**

***a. Body-Level Fields:***

1. tranid: Transaction ID (Sales Order number)
2. entity: Customer
3. trandate: Sales Order date
4. location: Location (Store or warehouse)
5. memo: Memo/Description

***b. Line-Level Fields:***

1. item: Item (product/service being sold)
2. quantity: Quantity ordered
3. rate: Rate per unit of the item
4. amount: Line item total amount
5. taxcode: Tax code for the item

***c. Sublist IDs:***

1. item: The item sublist for products/services ordered.
2. shipgroup: Shipping group sublist for multi-shipment sales orders.
3. taxdetails: tax details sublist for sales orders.

**C. Customer Record**

***a. Body-Level Fields:***

1. entityid: Customer ID/Name
2. email: Email address
3. phone: Phone number
4. category: Customer category
5. terms: Payment terms

***b. Line-Level Fields:***

1. addr1: Address Line 1
2. addr2: Address Line 2
3. city: City
4. state: State/Province
5. zip: Postal/ZIP Code

***c. Sublist IDs:***

1. addressbook: The address sublist for customer addresses.
2. contacts: Sublist for customer contacts.
3. salesrep: Sublist for sales representatives assigned to the customer.

**D. Vendor Record**

***a. Body-Level Fields:***

1. entityid: Vendor ID/Name
2. email: Email address
3. phone: Phone number
4. category: Vendor category
5. terms: Payment terms

***b. Line-Level Fields:***

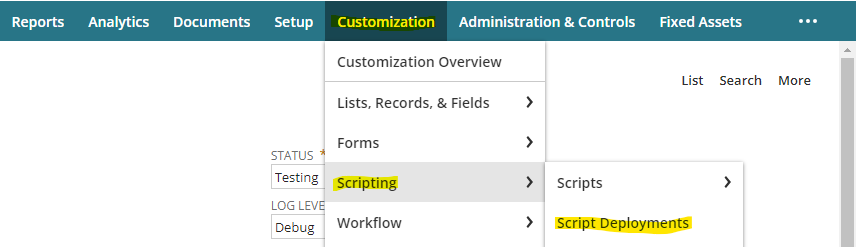
1. addr1: Address Line 1
2. addr2: Address Line 2
3. city: City
4. state: State/Province
5. zip: Postal/ZIP Code

***c. Sublist IDs:***

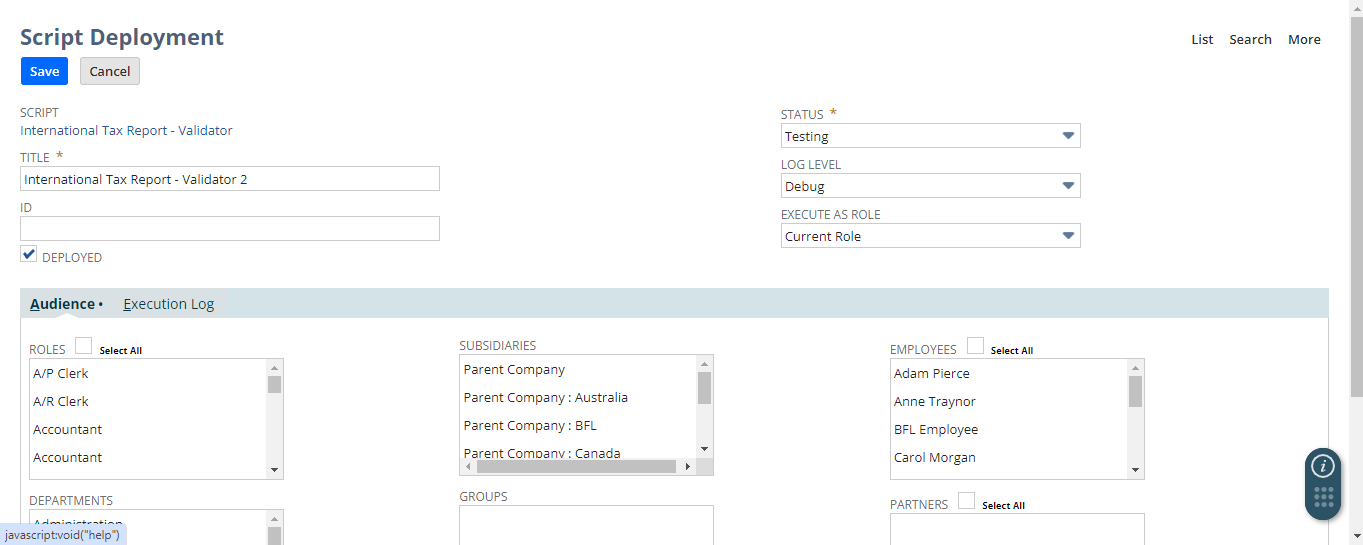
1. addressbook: The address sublist for vendor addresses.
2. contacts: Sublist for vendor contacts.
3. transactions: Sublist showing related transactions with the vendor.

**5. On the script deployment record write the purpose of each field mentioned.**

***Step 1* –** To navigate to script deployment, go to Customization -> Scripting -> Script Deployments



***Step 2 –*** Now you will find several fields such as: script, title, id, status, log level, execute as a role etc.



1. **SCRIPT** – Here you have to provide the script type such as Suitelet, RESTlet, User Event, Scheduled, Map/Reduce, Client
2. **TITLE**- Provide the title/name for the deployment using proper naming conventions.
3. **ID**- Here enter a custom internal ID for the script record, if left blank then the system will automatically generate an id for this field.
4. **DEPLOYED** - Click this checkbox if you want to deploy your script to run in NetSuite.
5. **STATUS** – It indicates whether the script is active or not.
   1. *Testing* – It is for pre-production purposes.
   2. *Released* – For production purposes.
6. **LOG LEVEL** – It determines the details of log recorded.
   1. *Debug* - Generally set when a script is in testing mode.
   2. *Audit* - Generally set for scripts running in production mode.
   3. *Error* – It is set during testing and production to capture any issue/error/failures.
   4. *Emergency* - Generally used for scripts running in production mode and to capture only server system failure.
7. **EXECUTE AS ROLE** - Select the role you want the script or plug-in implementation to run as. Accordingly, it will provide permission and restrictions for the executing script.