Week 5

Scheduled Scripts and the Script Debugger

Week 5 Overview

You will learn:

1. Scheduled Scripts
    a. Usage
    b. Scheduling Options
    c. Hands on Exercise
2. Script Debugger
    a. Usage
    b. Hands on Exercise

Introduction to Scheduled Scripts

What Are Scheduled Scripts

- Scheduled Scripts are server-side scripts in SuiteScript 2.x used to perform long-running, resource-intensive processes.

- They are not triggered by user actions but are executed based on a defined schedule or via manual initiation.

- Suitable for large data processing, batch operations, or periodic tasks

Usage of Scheduled Scripts

Examples

1. Generating bulk invoices
2. Syncing data with external systems
3. Updating large datasets in custom records

Scheduling options

How to schedule a script in NetSuite

1. Upload and deploy script
2. Change Status to Scheduled
3. Set Schedule
    a. Set Event (Single, Daily, Weekly, Monthly, Yearly)
    b. Specify Start Date
    c. Specify Start Time
    d. Specify how often the script is to repeat
    e. Set End Date (Or select No End Date)

Script Deployment

Save | Cancel

SCRIPT
Sample Scheduled Script

TITLE *
Sample Scheduled Script

ID
_eso_sample_ss

☑ DEPLOYED

1. STATUS *
Not Scheduled

LOG LEVEL
Debug

EXECUTE AS ROLE
Administrator

PRIORITY *
Standard

2. Schedule • Execution Log

- ◉ SINGLE EVENT
- ○ DAILY EVENT
- ○ WEEKLY EVENT
- ○ MONTHLY EVENT
- ○ YEARLY EVENT

3. START DATE *
12/16/2024

4. START TIME
6:00 pm

5. REPEAT

END BY

☑ NO END DATE

Save | Cancel

1. Status (Scheduled or Not Scheduled)
2. Event Frequency
3. Start Date
4. Start Time
5. Frequency (15 minutes, hours, etc..)

Introduction to the Script Debugger

What is the Script Debugger

- The Script Debugger allows developers to debug and troubleshoot scripts by running them step-by-step and monitoring their execution.

Script Debugger Features:

- Set breakpoints to pause script execution.
- Monitor variable values and logs in real-time.
- Step through code to identify logic or runtime errors.

Why Use the Script Debugger?

- Helps find and resolve issues quickly.
- Reduces time spent on manual testing and logging.

Prerequisites for Script Debugging

1. **NetSuite Account Access** (Sandbox preferred for testing).
2. **Script Debugger Permission**:
    a. Role must have permission for **SuiteScript Debugger**.
3. **Script Deployment**:
    a. The script to debug must be deployed and accessible.

Hands on exercise

We will do the hands-on exercise and then use this to help us understand the debugger!

Scenario:

Monthly Vendor Bill updates are uploaded to a folder in the File Cabinet. These files contain the approval status of existing Vendor Bills. Your scheduled script is to find the most recent file and update the related Vendor Bills.

See here for example CSV format

Solution:

Link to Solution :

https://github.com/DerekEsonus/SampleSuiteScripts/blob/main/SampleScheduledScript.js

Add the Folder and CSV to your file cabinet, update internal ID values in the CSV and in the script

Edit	1	HR Documents
Edit	-4	Images
Edit	118	Mail Template Image Folder
Edit	4	Sales Tools Tab
Edit	4180	Scheduled Script Sample Folder
Edit	117	Shipping Labels
Edit	-17	SSL Certificates

Scheduled Script Sample Folder

EDIT	INTERNAL ID	NAME ▲	SIZE	LAST MODIFIED
Edit	21128	12/17/2024 - ssDemoCsv	88 B	12/17/2024 7:08 am

For now we will set the Status to Not Scheduled so we can immediately execute the script

Script Deployment

Save | Cancel | Change ID | Actions

- Save & New
- Save & Copy
- Save and Execute

cript

Script

ID
customdeploy_eso_sample_schdscript_dply

☑ DEPLOYED

STATUS *
Not Scheduled

SEE INSTANCES
Status Page

LOG LEVEL
Debug

EXECUTE AS ROLE
Administrator

PRIORITY *
Standard

Schedule • Execution Log System Notes

- ◉ SINGLE EVENT
- ○ DAILY EVENT
- ○ WEEKLY EVENT
- ○ MONTHLY EVENT
- ○ YEARLY EVENT

START DATE *
12/17/2024

START TIME
6:00 pm

REPEAT

END BY
12/17/2024

☐ NO END DATE

Save | Cancel | Change ID | Actions

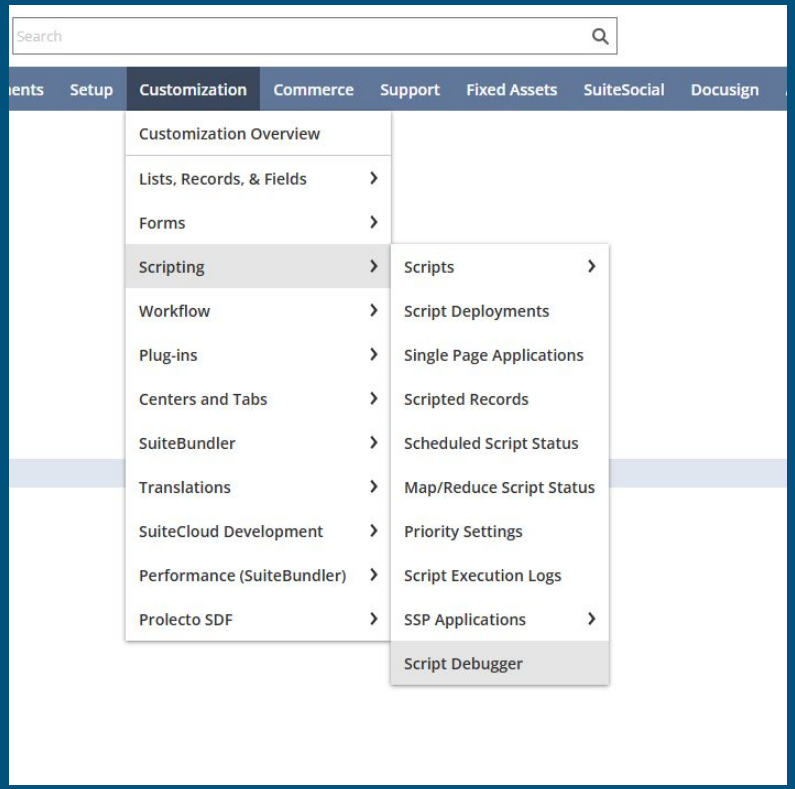
Using the Debugger

First, update the Deployment status to ‘Testing’

Using the debugger

Then, navigate to the debugger

Customization > Scripting > Script Debugger



Log into the debugger domain

Note - any actions your script does in the debugger will actually run on related records / fields in your NetSuite environment

Script Debugger

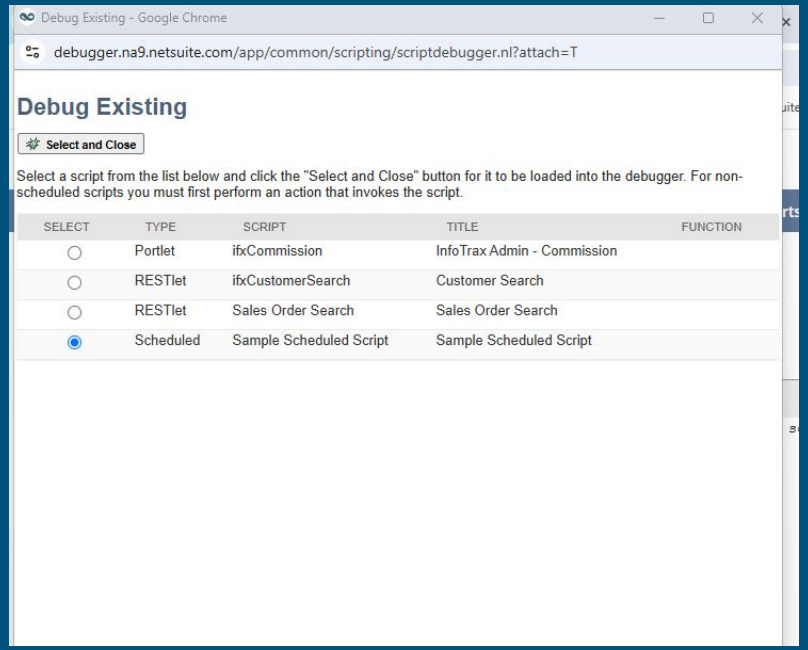
If you are accessing the Debugger domain through your production account, be aware that changes you make to your account on the Debugger domain will affect your production account.
If you are accessing the Debugger domain through a Beta or sandbox account, changes you make to your account on the Debugger domain will affect your Beta or sandbox account, respectively and will not affect your production account.

Click here to log in to the SuiteScript Debugger domain. Note that you will be logged off from your current session.

For steps on using the SuiteScript Debugger, see SuiteScript Debugger in the NetSuite Help Center.

Select the correct API version (usually 2.1) and click on Debug Existing

Select your script and click ‘Select and Close’



A new tab should open up with your script

/SuiteScripts/S...duledScript.js

```
/**
 * @NApiVersion 2.1
 * @NScriptType ScheduledScript
 */

define(['N/file', 'N/record', 'N/log', 'N/search'], function(file, record, log, search) {
    const execute = (context) => {
        log.debug('Scheduled Script', 'Starting Approval Status Update Process');

        try {
            // Step 1: Search for files in the folder
            const folderId = 4180; // Replace with your File Cabinet folder ID
            const files = [];

            const fileSearch = search.create({
                type: 'folder',
                filters: [['internalid', 'is', folderId]],
                columns: [
                    search.createColumn({ name: 'internalid', join: 'file' }),
                    search.createColumn({ name: 'name', join: 'file' })
                ]
            });

            fileSearch.run().getRange({ start: 0, end: 1000 }).forEach(result => {
                const fileName = result.getValue({ name: 'name', join: 'file' });
                const fileId = result.getValue({ name: 'internalid', join: 'file' });

                if (fileName.match(/\d{2}\/\d{2}\/\d{4}/)) { // Check for date pattern in filename
                    files.push({
                        id: fileId,
                        name: fileName,
                        created: new Date(fileName.match(/\d{2}\/\d{2}\/\d{4}/)[0])
                    });
                }
            });

            if (files.length === 0) {
                log.error('No Files Found', 'No valid Vendor Bill CSV files found in the folder.');
                return;
            }

            // Sort files by the date extracted from their names (most recent first)
            files.sort((a, b) => b.created - a.created);
            const recentFile = files[0];

            log.debug('Most Recent File', `File: ${recentFile.name}, ID: ${recentFile.id}`);

            // Step 2: Load the most recent file and parse its contents
            const paymentFile = file.load({ id: recentFile.id });
            const fileContents = paymentFile.getContents();

            const rows = fileContents.split('\n');
            rows.shift(); // Remove header row

            for (const row of rows) {
                if (!row.trim()) continue; // Skip empty rows
```

Debugger paused

Watch

Call Stack

:program
/SuiteScripts/S...ledScript.js:6

Scope

Local
this: Object global{Object:

Global

Breakpoints

/SuiteScripts/SampleScheduledS...
define(['N/file', 'N/recor...

/SuiteScripts/SampleScheduledSc...
fileSearch.run().getRange({

/SuiteScripts/S...duledScript.js

1. Breakpoints

2. Debug actions

```
/**
 * @NApiVersion 2.1
 * @NScriptType ScheduledScript
 */
define(['N/file', 'N/record', 'N/log', 'N/search'], function(file, record, log, search) {
    const execute = (context) => {
        log.debug('Scheduled Script', 'Starting Approval Status Update Process');

        try {
            // Step 1: Search for files in the folder
            const folderId = 4180; // Replace with your File Cabinet folder ID
            const files = [];

            const fileSearch = search.create({
                type: 'folder',
                filters: [['internalid', 'is', folderId]],
                columns: [
                    search.createColumn({ name: 'internalid', join: 'file' }),
                    search.createColumn({ name: 'name', join: 'file' })
                ]
            });

            fileSearch.run().getRange({ start: 0, end: 1000 }).forEach(result => {
                const fileName = result.getValue({ name: 'name', join: 'file' });
                const fileId = result.getValue({ name: 'internalid', join: 'file' });

                if (fileName.match(/\d{2}\/\d{2}\/\d{4}/)) { // Check for date pattern in filename
                    files.push({
                        id: fileId,
                        name: fileName,
                        created: new Date(fileName.match(/\d{2}\/\d{2}\/\d{4}/)[0])
                    });
                }
            });

            if (files.length === 0) {
                log.error('No Files Found', 'No valid Vendor Bill CSV files found in the folder.');
                return;
            }

            // Sort files by the date extracted from their names (most recent first)
            files.sort((a, b) => b.created - a.created);
            const recentFile = files[0];

            log.debug('Most Recent File', `File: ${recentFile.name}, ID: ${recentFile.id}`);

            // Step 2: Load the most recent file and parse its contents
            const paymentFile = file.load({ id: recentFile.id });
            const fileContents = paymentFile.getContents();

            const rows = fileContents.split('\n');
            rows.shift(); // Remove header row

            for (const row of rows) {
                if (!row.trim()) continue; // Skip empty rows
```

Debugger paused
Watch
Call Stack
:program
/SuiteScripts/S...ledScript.js:6
Scope
Local
this: Object global{Object: …
Global
Breakpoints
/SuiteScripts/SampleScheduledS…
define(['N/file', 'N/recor…
/SuiteScripts/SampleScheduledSc…
fileSearch.run().getRange({…

1. Add breakpoints (pause/stop points) by clicking on the line numbers

2. Debug Actions (from left to right)
   a. Continue (go to next breakpoint)
   b. Step Over
   c. Step Into
   d. Step Out
   e. Step

You can examine variable values in the console

The script is paused on the breakpoint on line 24.

Typing ‘fileSearch’ into the console will show you the value of the fileSearch const declared on line 24

Console Sources Memory Profiler

Filesystem Snippets

+ Add folder to workspace

Sync changes in DevTools with the local filesystem

Learn more

/SuiteScripts/S...duledScript.js

```
1  /**
2   * @NApiVersion 2.1
3   * @NScriptType ScheduledScript
4   */
5
6  define(['N/file', 'N/record', 'N/log', 'N/search'], function(file, record, log, search) {
7      const execute = (context) => {   context = ScheduledScriptContext {toString: ƒ}
8          log.debug('Scheduled Script', 'Starting Approval Status Update Process');
9
10         try {
11             // Step 1: Search for files in the folder
12             const folderId = 4180; // Replace with your File Cabinet folder ID
13             const files = [];
14
15             const fileSearch = search.create({
16                 type: 'folder',
17                 filters: [['internalid', 'is', folderId]],
18                 columns: [
19                     search.createColumn({ name: 'internalid', join: 'file' }),
20                     search.createColumn({ name: 'name', join: 'file' })
21                 ]
22             });
23
24             fileSearch.run().getRange({ start: 0, end: 1000 }).forEach(result => {
25                 const fileName = result.getValue({ name: 'name', join: 'file' });
26                 const fileId = result.getValue({ name: 'internalid', join: 'file' });
27
28                 if (fileName.match(/\d{2}\/\d{2}\/\d{4}/)) { // Check for date pattern in filename
29                     files.push({
30                         id: fileId,
31                         name: fileName,
32                         created: new Date(fileName.match(/\d{2}\/\d{2}\/\d{4}/)[0])
33                     });
34                 }
35             });
36
37             if (files.length === 0) {
38                 log.error('No Files Found', 'No valid Vendor Bill CSV files found in the folder.');
39                 return;
40             }
41
42             // Sort files by the date extracted from their names (most recent first)
43             files.sort((a, b) => b.created - a.created);
44             const recentFile = files[0];
45
46             log.debug('Most Recent File', `File: ${recentFile.name}, ID: ${recentFile.id}`);
47
48             // Step 2: Load the most recent file and parse its contents
49             const paymentFile = file.load({ id: recentFile.id });
50             const fileContents = paymentFile.getContents();
51
52             const rows = fileContents.split('\n');
53             rows.shift(); // Remove header row
54
55             for (const row of rows) {
56                 if (!row.trim()) continue; // Skip empty rows
57                 const fields = row.split(',');
58
59                 const recordId = fields[0].trim(); // Vendor Bill Internal ID
60                 const approvalStatus = fields[1].trim().toLowerCase(); // Approval Status (e.g., Approved
61
62                 // Only process records with "approved" status
63                 if (approvalStatus !== 'approved') {
64                     log.debug('Skipping Record', `Record ID: ${recordId}, Status: ${approvalStatus}`);
65                     continue;
66                 }
67
68                 try {
```

{} Line 24, Column 13

Console

Main Context Filter Default levels

```
Scheduled Script: Starting Approval Status Update Process
> fileSearch
<  ▸ NetSuiteObject {_load: ƒ, save: ƒ, run: ƒ, runPaged: ƒ, toString: ƒ, …}
>
```