



Session 8



Concurrency, Script Library, and
Repository Management



Session 8 Overview

- Organizing and managing scripts using script libraries
- Managing / monitoring concurrency
- Repository management: Version control and deployment strategies



Organizing and Managing Scripts Using Script Libraries



What Are Script Libraries?

Definition:

- Script libraries are a reusable script file that contains shared functions

Purpose:

- Avoid code duplication, improve maintainability

Examples:

- **Data validation:** ensure all new customer records meet business data integrity requirements
- **Logging and error handling:** standardized logging and error handling across multiple scripts

Benefits of Using Script Libraries

Code Reusability:

- Write once, use multiple times

Modularity:

- Keep scripts organized and structured.

Easier maintenance:

- Update logic in one place

Implementing Script Libraries

Steps:

1. Create a new SuiteScript File in the File Cabinet
2. Define reusable functions inside the script
3. Include the script in other scripts using `require()`
4. Call functions as needed

Creating Library

```
define(['N/record', 'N/search', 'N/log'], function(record, search, log) {

    function getCustomerEmail(customerId) {
        if (!customerId) return null;

        let customerRecord = record.load({
            type: record.Type.CUSTOMER,
            id: customerId
        });

        return customerRecord.getValue('email');
    }

    function validatePhoneNumber(phone) {
        var phoneRegex = /^\\d{10}$/; // Example: Only allows 10-digit numbers
        return phoneRegex.test(phone);
    }

    return {
        getCustomerEmail: getCustomerEmail,
        validatePhoneNumber: validatePhoneNumber
    };
});
```

```
define(['N/record', 'N/log', '/SuiteScripts/HelperFunctions'], function(record, log,
HelperFunctions) {
```

Using Library

1. Reference the file in your script define()
2. Use the functions in your script

```
function afterSubmit(context) {
    let newRecord = context.newRecord;
    let customerId = newRecord.getValue('entity');
    let transactionAmount = newRecord.getValue('total');

    // Use the Helper Library
    let customerEmail = HelperFunctions.getCustomerEmail(customerId);
    let formattedAmount = HelperFunctions.formatCurrency(transactionAmount);

    log.debug({
        title: 'Order Confirmation',
        details: `Customer Email: ${customerEmail}, Order Total:
${formattedAmount}`
    });


    // Log transaction details
    HelperFunctions.logTransactionDetails(newRecord.id);
}
return { afterSubmit: afterSubmit };
});
```


Hands-on Exercise



Objective: Create a script library and use it in a client script

Tasks:

1. Create a Script Library
2. Add utility functions (formatting, validation, etc..)
3. Reference it inside a script



Managing / Monitoring Concurrency



What is concurrency?

Definition:

- Multiple scripts executing simultaneously
- REST API request limits for your NetSuite environment

[Concurrency Governance cheat sheet](#)

SuiteScript governance rules

API Usage Limits:

- Scripts consume governance units

Governance Considerations:

- SSS_USAGE_LIMIT_EXCEEDED errors
- Exceeding concurrency may delay script execution

Issues caused by concurrent script executions

Performance Issues:

- Excessive concurrent executions can slow down the entire system

Governance Limit Errors:

- Exceeding limits can cause script failures

Monitoring Concurrency in NetSuite

Ways to Monitor:

- System notes & script execution logs
- NetSuite [concurrency monitoring](#) report

Hands-on Exercise

Objective: Get a real-time report on concurrency usage in your existing environment

Tasks:

- Access NetSuite's concurrency monitoring Dashboard
- Identify NetSuite concurrency usage

Bonus Task:

- Create a Saved search that displays server logs for NetSuite Scripts



Repository Management



Importance of Version Control

Why Version Control matters:

- Track changes over time
- Rollback to previous version when needed
- Enables team collaboration

Using Git for SuiteScript

Git Basics

Best Practices:

- Work in branches before merging
- Keep a backup of critical scripts (sandbox but ideally an external backup as well)

Deployment Strategies

NetSuite Deployment Methods:

1. Direct UPload to NetSuite File Cabinet
2. SuiteCloud Development Framework (SDF)

Third Party Tools:

- salto.io

Hands-On Exercise

Objective: Set up a Git repository for SuiteScript

Tasks:

- Create a Git repository
- Add a SuiteScript file
- Commit and push changes
- Experiment with branching and merging

Looking forward

The next session will cover:

- SuiteCloud Development Framework basics
- Deploying SuiteScripts using SDF