ORACLE NETSUITE

SuiteScript Developer Guide

2023.1

June 28, 2023

If this document is in private pre-General Availability status:

The information contained in this document is for informational sharing purposes only and should be considered in your capacity as a customer advisory board member or pursuant to your pre-General Availability trial agreement only. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described in this document may change and remains at the sole discretion of Oracle.

This document in any form, software or printed matter, contains proprietary information that is the exclusive property of Oracle. Your access to and use of this confidential material is subject to the terms and conditions of your Oracle Master Agreement, Oracle License and Services Agreement, Oracle PartnerNetwork Agreement, Oracle distribution agreement, or other license agreement which has been executed by you and Oracle and with which you agree to comply. This document and information contained herein may not be disclosed, copied, reproduced, or distributed to anyone outside Oracle without prior written consent of Oracle. This document is not part of your license agreement nor can it be incorporated into any contractual agreement with Oracle or its subsidiaries or affiliates.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Sample Code

Oracle may provide sample code in SuiteAnswers, the Help Center, User Guides, or elsewhere through help links. All such sample code is provided "as is" and “as available”, for use only with an authorized NetSuite Service account, and is made available as a SuiteCloud Technology subject to the SuiteCloud Terms of Service at www.netsuite.com/tos.

Oracle may modify or remove sample code at any time without notice.

No Excessive Use of the Service

As the Service is a multi-tenant service offering on shared databases, Customer may not use the Service in excess of limits or thresholds that Oracle considers commercially reasonable for the Service. If Oracle reasonably concludes that a Customer’s use is excessive and/or will cause immediate or ongoing performance issues for one or more of Oracle’s other customers, Oracle may slow down or throttle Customer’s excess use until such time that Customer’s use stays within reasonable limits. If Customer’s particular usage pattern requires a higher limit or threshold, then the Customer should procure a subscription to the Service that accommodates a higher limit and/or threshold that more effectively aligns with the Customer’s actual usage pattern.

Beta Features

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

Send Us Your Feedback

We'd like to hear your feedback on this document.

Answering the following questions will help us improve our help content:

- Did you find the information you needed? If not, what was missing?
- Did you find any errors?
- Is the information clear?
- Are the examples correct?
- Do you need more examples?
- What did you like most about this document?

Click here to send us your comments. If possible, please provide a page number or section title to identify the content you're describing.

To report software issues, contact NetSuite Customer Support.

Table of Contents

Setting Up Your SuiteScript Environment

Applies to: SuiteScript 2.x | SuiteCloud Developer

Before working with SuiteScript, you should configure both your NetSuite account and your SuiteScript development environment. To do so, see the following sections:

- Environment Setup Overview
- Configuring NetSuite for SuiteScript
- Working with IDEs Other Than SuiteCloud IDEs

Environment Setup Overview

To configure your SuiteScript development environment, see the following topics:

- Installing and Setting Up SuiteCloud Extension for Visual Studio Code
- Installing and Setting Up SuiteCloud IDE Plug-in for WebStorm

Configuring NetSuite for SuiteScript

You must complete all of the following tasks to enable SuiteScript in your account and to access the internal record and field IDs that may be required as parameter values in your SuiteScript code. These tasks include:

1. Enabling SuiteScript
2. Showing Record and Field IDs in Your Account
3. Setting Roles and Permissions for SuiteScript

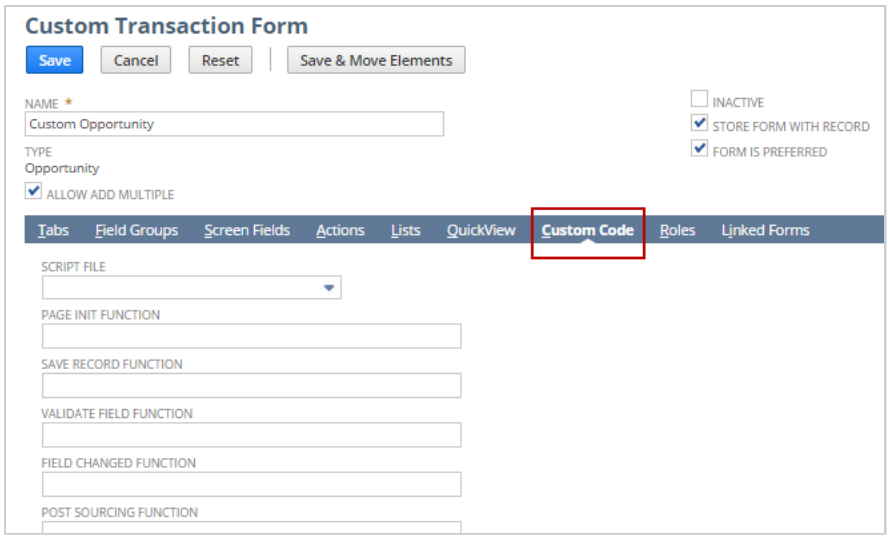
Enabling SuiteScript

Before you can run SuiteScript in your NetSuite account, an Account Administrator must enable the SuiteScript feature.

To enable SuiteScript:

1. Go to Setup > Company > Enable Features.
2. Click the **SuiteCloud** tab.
3. Under SuiteScript, click the **Client SuiteScript** or **Server SuiteScript** box (or both, depending on the scripts you want to run).
4. Click **Save**.

Note: If Client SuiteScript is enabled, the **Custom Code** tab becomes available on entry and transactions forms (see figure below). Here you select which client scripts to associate with the **current form**. For information about attaching client scripts to NetSuite forms, see the help topic SuiteScript 2.x Hello World. For information about entry and transactions forms, see the help topic Custom Forms in the SuiteBuilder (Customization) Guide.



Showing Record and Field IDs in Your Account

After enabling the SuiteScript feature, you should enable the **Show Internal IDs** preference. Enabling this preference lets you see the internal IDs for all fields and records in NetSuite.

When referencing a specific record or field in SuiteScript, you use it's internal ID. Even if the record or field's UI label is changed, the internal ID will remain constant.

After enabling **Show Internal IDs**, see the following sections for steps on viewing the IDs from within NetSuite:

- How to Find a Record's Internal ID
- How to Find a Field's Internal ID

To show internal NetSuite IDs:

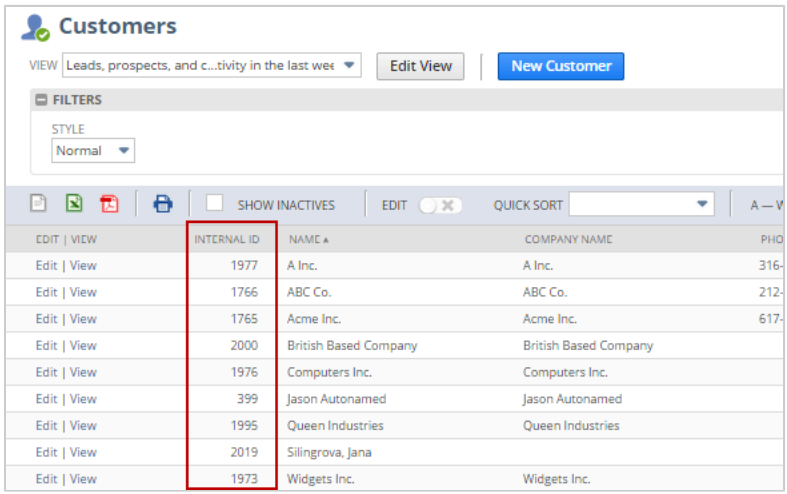
1. Go to Home > Set Preferences.
2. Click the **General** tab and then click the **Show Internal IDs** box.
3. Click **Save**.

For examples of how internal IDs are referenced in the SuiteScript API, see the help topic record.load(options) or search.load(options). Also note that when writing SuiteScript, **all record and field IDs must be in lowercase**.

How to Find a Record's Internal ID

A record's internal ID is unique and associated with the record at the time it is created. After the **Show Internal IDs** preference is enabled, the internal IDs for each record are displayed in the Internal ID column of record lists (see figure below).

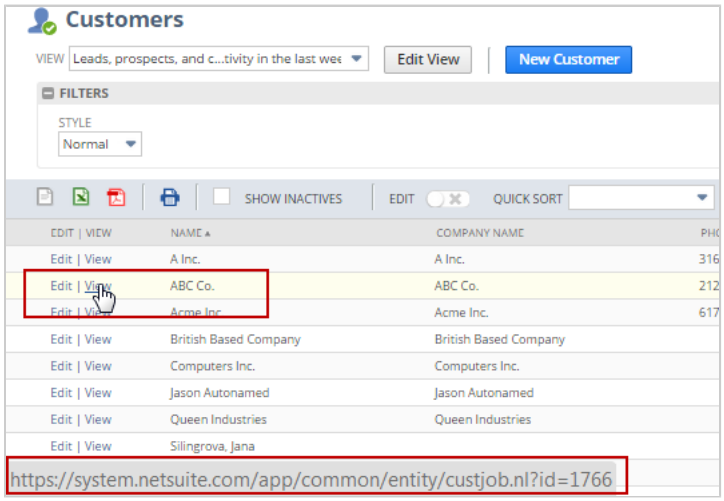
For example, to see an internal ID for a specific customer record, go to Lists > Relationships > Customers (Administrator). In the Internal ID column, the internal ID appears next to each record in the Customers list (see figure below).



See for steps on enabling the **Show Internal IDs** preference.

If the **Show Internal IDs** preference is NOT enabled, or if the internal IDs are not displayed on a particular page within NetSuite, you can see the internal ID for a record by hovering over a link to that record. The internal ID is displayed as a parameter in the URL in the browser status bar.

The following figure shows that if you hover over a link to the ABC Co. customer record, the internal record ID (1766) appears in the browser status bar.



Tip: You obtain the internal ID of a record type (for example, 'salesorder') by going to the SuiteScript Records Browser. For information about using the SuiteScript Records Browser, see Working with the SuiteScript Records Browser in the NetSuite Help Center.

How to Find a Field's Internal ID

Internal field IDs must be used when calling a field from a SuiteScript API.

Tip: To determine if **Show Internal IDs** is enabled, see the help topic Setting the Show Internal IDs Preference.

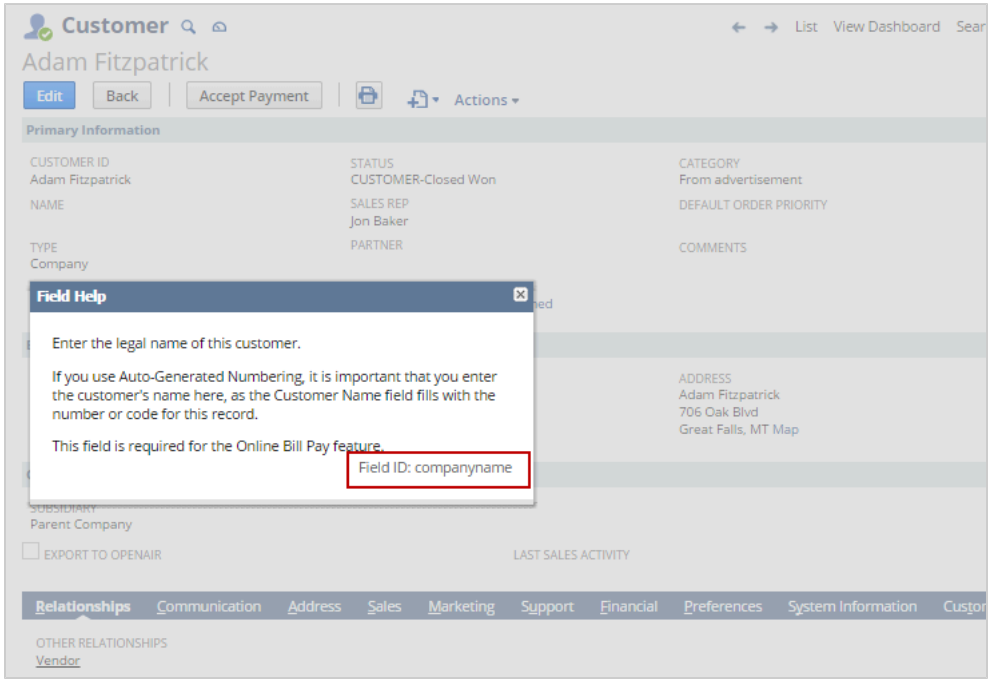
There are multiple ways to view a field's internal ID:

- Working with the SuiteScript Records Browser
- Internal IDs for Standard and Custom Fields in Field Level Help Window
- Internal IDs for Custom Fields on List Pages

Some fields are not displayed to users in NetSuite. Use the SuiteScript Records Browser to find internal IDs for those fields.

Internal IDs for Standard and Custom Fields in Field Level Help Window

With Show Internal IDs enabled, you can also view internal field IDs for both standard and custom fields, by clicking the field label in the UI. The figure below shows the Field Level Help window that opens when a field label is clicked, in this case, the Company label. The internal ID for the Company field is `companyname`, which appears in the bottom-right corner of the Field Level Help window.

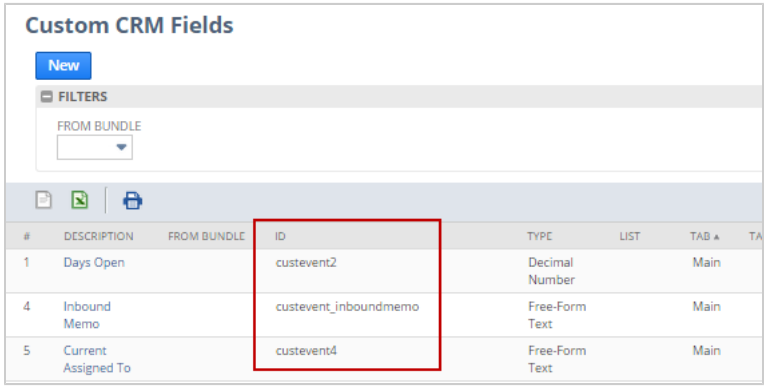


Note: When creating custom fields, you can specify your own field ID, or you can accept the default ID assigned by NetSuite. To ensure that the field IDs make sense in the context of your business environment, you should define your own custom field IDs. For detailed information about creating custom fields and assigning custom field IDs, refer to Custom Fields in the NetSuite Help Center.

Internal IDs for Custom Fields on List Pages

When the Show Internal IDs preference is enabled, internal IDs for each custom field are displayed in the Internal ID column of a custom field page. For example, to see the internal IDs for custom CRM fields, go

to Customization > Lists, Records, & Fields > CRM Fields (Administrator). The ID column appears in the list of custom fields, as shown in the figure.



Setting Roles and Permissions for SuiteScript

NetSuite provides many standard roles with predefined permissions. A role (and its set of predefined permissions) lets customers, vendors, partners, and employees access specific areas of your data. Each role grants access at a certain level for each permission. Use the Administrator role for complete access to all SuiteScript functionality, or use the permissions in the table to allow more granular access.

For additional information about roles and permissions, see the help topic NetSuite Users & Roles.

To add SuiteScript permission:

1. Go to Setup > Users/Roles > Manage Roles
2. Click **Edit** to make changes to a role.
3. In the **Permissions** tab, click **Setup** and add the SuiteScript permission.

Note: If you customize a role to add SuiteScript functionality, you must also include the permissions to customize entry forms and transaction forms.

SuiteScipt permissions

Action	Permission Required
View script records and script deployment records.	Role with SuiteScript permission (View level)
Create and view script records and script deployment records.	Role with SuiteScript permission (Create level)
Create, view, and edit script records and script deployment records.	Role with SuiteScript permission (Edit level)
Use **Save and Execute** from the script deployment record UI to execute an on-demand scheduled script or an on-demand map/reduce script.	Role with SuiteScript permission (Full level)
Debug a SuiteScript 1.0, SuiteScript 2.0, or SuiteScript 2.x script.	Role with SuiteScript permission (Full level)

Action	Permission Required
Use an API (for example, ScheduledScriptTask.submit()) to execute an on-demand scheduled script or an on-demand map/reduce script.	Role with SuiteScript Scheduling permission (Full level)
Execute a script with one of the following APIs: ■ SuiteScript 1.0 (see the help topic SuiteScript 1.0 Documentation) □ `nlobjContext.getPercentageComplete()` □ `nlobjContext.setPercentageComplete(pct)` ■ SuiteScript 2.x □ Script.percentComplete	Role with SuiteScript Scheduling permission (Full level)

For additional information about roles and permissions, see the help topic NetSuite Users & Roles.

After completing the SuiteScript configuration, you will need to continue to set up your IDE. To help you set up your IDE, see the following help topics:

- SuiteCloud Extension for Visual Studio Code Overview
- SuiteCloud IDE Plug-in for WebStorm Overview
- Working with IDEs Other Than SuiteCloud IDEs

Working with IDEs Other Than SuiteCloud IDEs

Although you should use the SuiteCloud IDEs when writing SuiteScript, you can still use other development tools to create SuiteScript. Note, however, without the SuiteCloud IDEs, you will not be able to automatically upload SuiteScript files into the NetSuite File Cabinet.

For more information about additional development environments, see the help topic SuiteCloud Extension for Visual Studio Code Overview or SuiteCloud IDE Plug-in for WebStorm Overview.

If you choose to use a development tool or IDE other than the SuiteCloud IDEs, see the following sections:

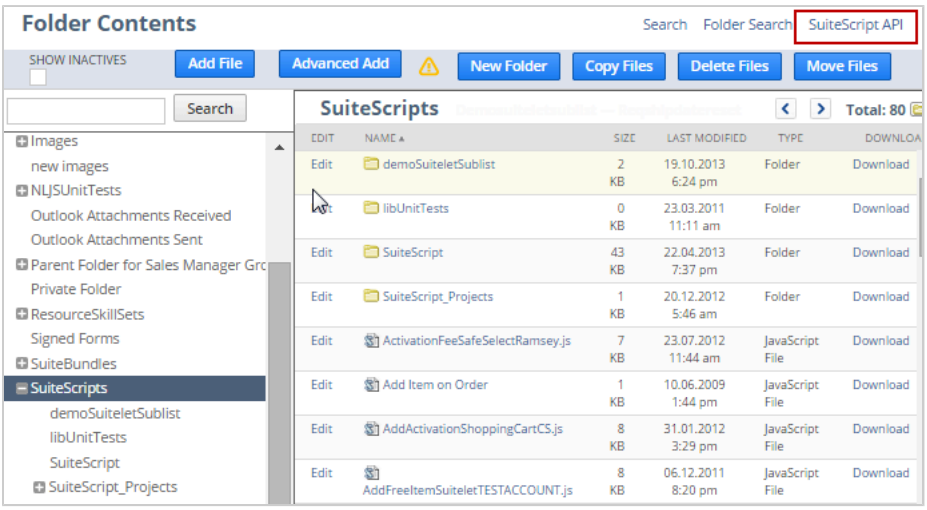
- Adding the SuiteScript Library File to Your IDE
- Uploading SuiteScript into the File Cabinet Without SuiteCloud IDEs

Adding the SuiteScript Library File to Your IDE

If you are working with an IDE other than the SuiteCloud IDEs, you should still add the SuiteScript library file to your SuiteScript project folder in your IDE.

To add the SuiteScript library file:

1. In NetSuite, go to Documents > Files > SuiteScripts.
2. Next, click the link to the **SuiteScript API** file (see figure).



3. Copy and paste the SuiteScript API file into your IDE.
4. Save the file as a .js file.

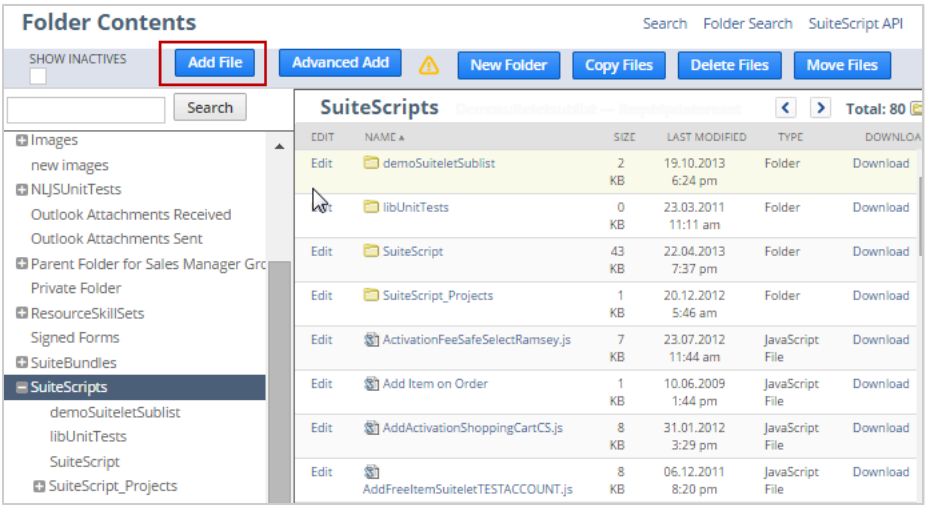
Uploading SuiteScript into the File Cabinet Without SuiteCloud IDEs

After the SuiteScript feature is enabled, a new SuiteScripts folder is created in the NetSuite File Cabinet. The File Cabinet is considered as the central repository for all your .js SuiteScript files. Therefore, your SuiteScript files should exist in the File Cabinet before they can be executed in your NetSuite account. The **SuiteScripts** folder within the File Cabinet is provided for convenience, however, you can store the script files in any location.

If you are not uploading your scripts into the File Cabinet using the SuiteCloud IDEs, use the following steps:

To upload SuiteScript into the File Cabinet:

1. Go to Documents > Files > SuiteScripts.
2. Click the **Add File** button.



3. In the File Upload window that appears, select the file you want to upload and click **Open**.

Note: If you make changes to a SuiteScript file that already exists in the File Cabinet, follow steps 1–3 to re-upload the changed file. Click **OK** to overwrite the previous file and load your changes.

SuiteScript Governance and Limits

Applies to: SuiteScript 2.x | SuiteCloud Developer

To optimize application performance, NetSuite uses a SuiteScript governance model based on usage units. If the number of allowable usage units is exceeded, the script execution is terminated.

Usage units are tracked on two levels: the script type level and the API level. Each script type can execute a system-defined number of usage units, and each SuiteScript API can consume a system-defined number of usage units.

The character limit for keys in map/reduce scripts (specifically, in mapContext or reduceContext objects) is 3,000 characters. In addition, error messages are returned when a key is longer than 3,000 characters or a value is larger than 10 MB. Keys longer than 3,000 characters will return the error KEY_LENGTH_IS_OVER_3000_BYTES. Values larger than 10 MB will return the error VALUE_LENGTH_IS_OVER_10_MB.

If you have map/reduce scripts that use the mapContext.write(options) or reduceContext.write(options) methods, make sure that key strings are shorter than 3,000 characters and value strings are smaller than 10 MB. Make sure that you consider the potential length of any dynamically generated strings, which may exceed these limits. You should also avoid using keys, instead of values, to pass your data.

The governance limits for each SuiteScript 2.x API are included in individual SuiteScript 2.x method topics. Methods are organized by modules, and all modules are listed in the help topic SuiteScript 2.x Modules. A summary table is also available in this Developer Guide (see SuiteScript 2.x API Governance).

Important: SuiteScript thresholds are based on the volume of activity that a company's users can manually generate. However, automated functions that generate excessive levels of activity may trigger metering of script execution as referenced in the NetSuite Main Terms of Service (TOS).

See the following help topics to learn about usage unit governance as it applies to specific script types, individual APIs, monitoring, and logging:

- Script Type Usage Unit Limits
- SuiteScript 2.x API Governance
- Monitoring Script Usage
- Governance on Script Logging

Related to governance, limits are enforced on certain aspects of script execution. When you run a search using the N/search module, the number of search results you receive is limited. There are also limits on the amount of time a script can run, based on the script type. See the following help topics to learn about these limits:

- Search Result Limits
- Script Execution Time Limits

Important: NetSuite uses internal mechanisms to detect scripts that may include infinite loops when executing. When they occur, script execution is terminated and an `SSS_INSTRUCTION_COUNT_EXCEEDED` error message is thrown. If you receive this error, you should examine all of the execution loops in your script to ensure that they contain either a terminating condition or a condition that can be met.

Script Type Usage Unit Limits

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following table lists the maximum allowable usage units for each SuiteScript (1.0 and 2.x) script type. You can use the Script.getRemainingUsage() method to see how many usage units you have remaining for a particular script.

Script Type	Total usage units Allowed per Script	Notes
Bundle Installation Scripts (SuiteScript 1.0) SuiteScript 2.x Bundle Installation Script Type	10,000	The limit is 10,000 usage units per execution.
Client Scripts (SuiteScript 1.0) SuiteScript 2.x Client Script Type	1,000	Client scripts are metered on a per-script basis. If an account has one form-level client script attached to a form and one record-level client script deployed to a record (which may be associated with a form), each client script can total 1,000 usage units. Usage units are not shared by all the client scripts associated with a form or record. For information about record- and form-level client scripts, see the help topic Record-Level and Form-Level Script Deployments.
SuiteScript 2.x Map/Reduce Script Type	—	Map/reduce scripts are available only in SuiteScript 2.x. There are no limits imposed on the full duration of a map/reduce script deployment instance. Instead, isolated components of the deployment, such as the usage units used by a single method invocation, are regulated. For more information, see the help topic Map/Reduce Governance.
Mass Update Scripts (SuiteScript 1.0) SuiteScript 2.x Mass Update Script Type	1,000	The limit is 1,000 usage units per record or execution of the script.
Portlet Scripts (SuiteScript 1.0) SuiteScript 2.x Portlet Script Type	1,000	—
RESTlets (SuiteScript 1.0) SuiteScript 2.x RESTlet Script Type	5,000	The SuiteScript governance model for RESTlets tracks usage units on the API level and the script level. For more information, see the help topic RESTlet Governance

Script Type	Total usage units Allowed per Script	Notes
Scheduled Scripts (SuiteScript 1.0) SuiteScript 2.x Scheduled Script Type	10,000	Within one scheduled script, all actions combined cannot exceed 10,000 usage units. For example, a scheduled script that includes two calls to record.transform(options) and one call to email.send(options) consumes from 24 to 40 usage units (depending on the record type for record.transform(options)) out of a possible 10,000 usage units available. If you have a scheduled script with potentially long execution times, you should consider using a map/reduce script instead. SuiteScript 2.x does not have a method to allow you to set recovery points or provide a yield to avoid exceeding the allowed governance for a scheduled script. A map/reduce script has built-in yielding and can be submitted for processing in the same ways as a scheduled script.
SuiteScript 2.x SDF Installation Script Type	10,000	The limit is 10,000 usage units per execution.
Suitelets (SuiteScript 1.0) SuiteScript 2.x Suitelet Script Type	1,000	Within one Suitelet, all actions combined cannot exceed 1,000 usage units. For example, a Suitelet that calls record.create(options) and http.get(options) consumes from 12 to 20 usage units (depending on the record type for record.create(options)) out of a possible 1,000 usage units available. Regardless of the 1,000 unit limit for Suitelets, you should create your Suitelets to be responsive to users, otherwise user experience may be impacted.
User Event Scripts (SuiteScript 1.0) SuiteScript 2.x User Event Script Type	1,000	Regardless of the 1,000 usage unit limit for user event scripts, you should create your scripts so that they are responsive to users, otherwise user experience may be impacted.
Workflow Action Scripts (SuiteScript 1.0) SuiteScript 2.x Workflow Action Script Type (also referred to as Custom Action in SuiteFlow)	1,000	Within one workflow state, all actions combined cannot exceed 1,000 usage units. For example, if you have developed a custom action (using a workflow action script) that consumes 990 usage units, be aware of the unit consumption of the other actions within that state.
Core Plug-ins	Varies based on the plug-in	The default limit for core plug-ins that do not have more restrictive limits defined is 10,000. See the help topic for each core plug-in for any specific usage unit limits.
Custom Plug-ins	10,000	The limit is 10,000 usage units per plug-in.
SSP Application Scripts	1,000	For more information, see the help topic SSP Application Governance.

Note: There is also a limit of 1,000 usage units when using the SuiteScript Debugger. For more information, see Script Debugger Metering and Permissions.

SuiteScript 2.x API Governance

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following table shows the governance usage units used for each SuiteScript 2.x method. You can use the Script.getRemainingUsage() method to see how many usage units you have remaining for a particular script.

The governance model takes into account the NetSuite processing requirements for three categories of records: custom records, standard transaction records, and standard non-transaction records. For example, custom records require less processing than standard records, therefore, the usage unit cost for custom records is lower than for standard records. Similarly, standard non-transaction records require less processing than standard transaction records, therefore, the usage unit cost for standard non-transaction records is lower than for standard transaction records. Standard transaction records include records such as cash refund, customer deposit, and item fulfillment. Standard non-transaction records include records such as activity, inventory item, and customer. You can see an example of this in the N/record module methods listed below.

Record categories are included in the SuiteScript Supported Records help topic. Record types categorized as activity, communication, customization, entity, file cabinet, item, list, marketing, subrecord, support, or website are considered to be standard non-transaction records.

For examples of API governance, see API Governance Examples.

For tips on how to monitor how many usage units your script is using, see Monitoring Script Usage.

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
N/action Module	Action(options) Action.promise(options)	None
	Action.execute(options) Action.execute.promise(options)	None
	Action.executeBulk(options)	50
	action.getBulkStatus(options)	None
	action.execute(options) action.execute.promise(options)	None
	action.executeBulk(options)	50
	action.find(options) action.find.promise(options)	None
	action.get(options) action.get.promise(options)	None
N/auth Module	auth.changeEmail(options)	10
	auth.changePassword(options)	10
N/cache Module	Cache.get(options)	1 if the value is present in the cache

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
		2 if the loader function is used
	Cache.put(options)	1
	Cache.remove(options)	1
	cache.getCache(options)	None
N/certificateControl Module	Certificate.save()	10
	certificateControl.createCertificate(options)	10
	certificateControl.deleteCertificate(options)	10
	certificateControl.findCertificates(options)	10
	certificateControl.findUsages(options)	10
	certificateControl.loadCertificate(options)	10
N/commerce Modules	recordView.viewItems(options)	None
	recordView.viewWebsite(options)	None
N/compress Module	Archiver.add(options)	None
	Archiver.archive(options)	None
	compress.createArchiver()	None
	compress.gzip(options)	None
	compress.gunzip(options)	None
N/config Module	config.load(options)	10
N/crypto Module	Cipher.final(options)	None
	Cipher.update(options)	None
	Decipher.final(options)	None
	Decipher.update(options)	None
	Hash.digest(options)	None
	Hash.update(options)	None
	Hmac.digest(options)	None
	Hmac.update(options)	None
	crypto.createCipher(options)	None
	crypto.createDecipher(options)	None
	crypto.createHash(options)	None
	crypto.createHmac(options)	None
	crypto.createSecretKey(options)	None
N/crypto/certificate Module	SignedXml.asFile()	None
	SignedXml.asString()	None
	SignedXml.asXml()	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	Signer.sign(options)	None
	Signer.update(options)	None
	Verifier.update(options)	None
	Verifier.verify(options)	None
	certificate.createSigner(options)	10
	certificate.createVerifier(options)	10
	certificate.verifyXmlSignature(options)	10
	certificate.signXml(options)	10
N/currency Module	currency.exchangeRate(options)	10
N/currentRecord Module	CurrentRecord.cancelLine(options)	None
	CurrentRecord.commitLine(options)	None
	CurrentRecord.findMatrixSublistLineWithValue (options)	None
	CurrentRecord.findSublistLineWithValue (options)	None
	CurrentRecord.getCurrentMatrixSublistValue (options)	None
	CurrentRecord.getCurrentSublistIndex (options)	None
	CurrentRecord.getCurrentSublistSubrecord (options)	None
	CurrentRecord.getCurrentSublistText(options)	None
	CurrentRecord.getCurrentSublistValue (options)	None
	CurrentRecord.getField(options)	None
	CurrentRecord.getLineCount(options)	None
	CurrentRecord.getMatrixHeaderCount (options)	None
	CurrentRecord.getMatrixHeaderField(options)	None
	CurrentRecord.getMatrixHeaderValue(options)	None
	CurrentRecord.getMatrixSublistField(options)	None
	CurrentRecord.getMatrixSublistValue(options)	None
	CurrentRecord.getSublist(options)	None
	CurrentRecord.getSublistField(options)	None
	CurrentRecord.getSublistText(options)	None
	CurrentRecord.getSublistValue(options)	None
	CurrentRecord.getSubrecord(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	CurrentRecord.getText(options)	None
	CurrentRecord.getValue(options)	None
	CurrentRecord.hasCurrentSublistSubrecord (options)	None
	CurrentRecord.hasSublistSubrecord(options)	None
	CurrentRecord.hasSubrecord(options)	None
	CurrentRecord.insertLine(options)	None
	CurrentRecord.removeCurrentSublist Subrecord(options)	None
	CurrentRecord.removeLine(options)	None
	CurrentRecord.removeSubrecord(options)	None
	CurrentRecord.selectLine(options)	None
	CurrentRecord.selectNewLine(options)	None
	CurrentRecord.setCurrentMatrixSublistValue (options)	None
	CurrentRecord.setCurrentSublistText(options)	None
	CurrentRecord.setCurrentSublistValue (options)	None
	CurrentRecord.setMatrixHeaderValue(options)	None
	CurrentRecord.setMatrixSublistValue(options)	None
	CurrentRecord.setText(options)	None
	CurrentRecord.setValue(options)	None
	Field.getSelectOptions(options)	None
	Field.insertSelectOption(options)	None
	Field.removeSelectOption(options)	None
	Sublist.getColumn(options)	None
	currentRecord.get() currentRecord.get.promise()	None
N/dataset Module	Dataset.getExpressionFromColumn(options)	None
	Dataset.run()	10
	Dataset.runPaged()	10
	Dataset.save(options)	10
	dataset.create(options)	None
	dataset.createColumn(options)	None
	dataset.createCondition(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	dataset.createJoin(options)	None
	dataset.createTranslation(options)	None
	dataset.describe(options)	10
	dataset.list()	10
	dataset.listPaged(options)	10
	dataset.load(options)	10
N/datasetLink Module	datasetLink.create(options)	None
N/email Module	email.send(options) email.send.promise(options)	20
	email.sendBulk(options) email.sendBulk.promise(options)	10
	email.sendCampaignEvent(options) email.sendCampaignEvent.promise(options)	10
N/encode Module	encode.convert(options)	None
N/error Module	error.create(options)	None
N/file Module	File.appendLine(options)	None
	File.getContents()	None
	File.getReader()	None
	File.getSegments(options)	None
	File.appendLine(options)	None
	File.lines.iterator()	None
	File.resetStream()	None
	File.save()	20
	Reader.readChars(options)	None
	Reader.readUntil(options)	None
	file.create(options)	None
	file.delete(options)	20
	file.load(options)	10
N/format Module	format.format(options)	None
	format.parse(options)	None
N/format/i18n Module	CurrencyFormatter.format(options)	10
	NumberFormatter.format(options)	10
	PhoneNumberFormatter.format(options)	None
	PhoneNumberParser.parse(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	format.getCurrencyFormatter(options)	10
	format.getNumberFormatter(options)	10
	format.spellOut(options)	None
N/http Module	ServerRequest.getLineCount(options)	None
	ServerRequest.getSublistValue(options)	None
	ServerResponse.addHeader(options)	None
	ServerResponse.getHeader(options)	None
	ServerResponse.renderPdf(options)	10
	ServerResponse.sendRedirect(options)	None
	ServerResponse.setCdnCacheable(options)	None
	ServerResponse.setHeader(options)	None
	ServerResponse.write(options)	None
	ServerResponse.writeFile(options)	None
	ServerResponse.writeLine(options)	None
	ServerResponse.writePage(options)	None
	http.get(options) http.get.promise(options)	10
	http.delete(options) http.delete.promise(options)	10
	http.post(options) http.post.promise(options)	10
	http.put(options) http.put.promise(options)	10
	http.request(options) http.request.promise(options)	10
N/https Module	SecureString.appendSecureString(options)	None
	SecureString.appendString(options)	None
	SecureString.convertEncoding(options)	None
	SecureString.hash(options)	None
	SecureString.hmac(options)	None
	ServerRequest.getLineCount(options)	None
	ServerRequest.getSublistValue(options)	None
	ServerResponse.addHeader(options)	None
	ServerResponse.getHeader(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	ServerResponse.renderPdf(options)	10
	ServerResponse.sendRedirect(options)	None
	ServerResponse.setCdnCacheable(options)	None
	ServerResponse.setHeader(options)	None
	ServerResponse.write(options)	None
	ServerResponse.writeFile(options)	None
	ServerResponse.writeLine(options)	None
	ServerResponse.writePage(options)	10
	https.createSecretKey(options) https.createSecretKey.promise(options)	None
	https.createSecureString(options) https.createSecureString.promise(options)	None
	https.get(options) https.get.promise(options)	10
	https.delete(options) https.delete.promise(options)	10
	https.post(options) https.post.promise(options)	10
	https.put(options) https.put.promise(options)	10
	https.request(options) https.request.promise(options)	10
	https.requestRestlet(options)	10
	https.requestSuiteTalkRest(options)	10
N/https/clientCertificate Module	clientCertificate.delete(options)	10
	clientCertificate.get(options)	10
	clientCertificate.post(options)	10
	clientCertificate.put(options)	10
	clientCertificate.request(options)	10
N/keyControl Module	Key.save()	10
	keyControl.createKey(options)	10
	keyControl.deleteKey(options)	10
	keyControl.findKeys(options)	10
	keyControl.loadKey(options)	10

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
N/log Module	log.audit(options)	Amount of logging in any 60–minute period is limited. See Governance on Script Logging.
	log.debug(options)	
	log.emergency(options)	
	log.error(options)	
N/piremoval Module	PiRemovalTask.deleteTask()	20
	PiRemovalTask.run()	20
	PiRemovalTask.save()	20
	piremoval.createTask(options)	None
	piremoval.deleteTask(options)	20
	piremoval.getTaskStatus(options)	None
	piremoval.loadTask(options)	None
N/plugin Module	plugin.findImplementations(options)	None
	plugin.loadImplementation(options)	None
N/portlet Module	portlet.refresh()	None
	portlet.resize()	None
N/query Module	Component.autoJoin(options)	None
	Component.createColumn(options)	None
	Component.createCondition(options)	None
	Component.createSort(options)	None
	Component.join(options)	None
	Component.joinFrom(options)	None
	Component.joinTo(options)	None
	PagedData.iterator()	None
	Query.and(conditions)	None
	Query.autoJoin(options)	None
	Query.createColumn(options)	None
	Query.createCondition(options)	None
	Query.createSort(options)	None
	Query.join(options)	None
	Query.joinFrom(options)	None
	Query.joinTo(options)	None
	Query.run() Query.run.promise()	10
	Query.runPaged()	10

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	Query.runPaged.promise()	
	Query.not(condition)	None
	Query.or(conditions)	None
	Query.toSuiteQL()	None
	Result.asMap()	None
	ResultSet.asMappedResults()	None
	ResultSet.iterator()	None
	SuiteQL.run()	10
	SuiteQL.runPaged(options)	10
	query.create(options)	None
	query.createPeriod(options)	None
	query.createRelativeDate(options)	None
	query.delete(options)	5
	query.listTables(options)	None
	query.load(options) query.load.promise(options)	5
	query.runSuiteQL(options)	10
	query.runSuiteQLPaged(options)	10
N/record Module	Field.getSelectOptions(options)	None
	Macro(options) Macro.promise(options)	None
	Macro.execute(options) Macro.execute.promise(options)	None
	Record.cancelLine(options)	None
	Record.commitLine(options)	None
	Record.executeMacro(options) Record.executeMacro.promise(options)	None
	Record.findMatrixSublistLineWithValue(options)	None
	Record.findSublistLineWithValue(options)	None
	Record.getCurrentMatrixSublistValue(options)	None
	Record.getCurrentSublistField(options)	None
	Record.getCurrentSublistIndex(options)	None
	Record.getCurrentSublistSubrecord(options)	None
	Record.getCurrentSublistText(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	Record.getCurrentSublistValue(options)	None
	Record.getField(options)	None
	Record.getFields()	None
	Record.getLineCount(options)	None
	Record.getMacro(options)	None
	Record.getMacros()	None
	Record.getMatrixHeaderCount(options)	None
	Record.getMatrixHeaderField(options)	None
	Record.getMatrixHeaderValue(options)	None
	Record.getMatrixSublistField(options)	None
	Record.getMatrixSublistValue(options)	None
	Record.getSublist(options)	None
	Record.getSublists()	None
	Record.getSublistField(options)	None
	Record.getSublistFields(options)	None
	Record.getSublistSubrecord(options)	None
	Record.getSublistText(options)	None
	Record.getSublistValue(options)	None
	Record.getSubrecord(options)	None
	Record.getText(options)	None
	Record.getValue(options)	None
	Record.hasCurrentSublistSubrecord(options)	None
	Record.hasSublistSubrecord(options)	None
	Record.hasSubrecord(options)	None
	Record.insertLine(options)	None
	Record.removeCurrentSublistSubrecord(options)	None
	Record.removeLine(options)	None
	Record.removeSublistSubrecord(options)	None
	Record.removeSubrecord(options)	None
	Record.save(options) Record.save.promise(options)	20 for transaction records 4 for custom records 10 for all other records
	Record.selectLine(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	Record.selectNewLine(options)	None
	Record.setCurrentMatrixSublistValue(options)	None
	Record.setCurrentSublistText(options)	None
	Record.setCurrentSublistValue(options)	None
	Record.setMatrixHeaderValue(options)	None
	Record.setMatrixSublistValue(options)	None
	Record.setSublistText(options)	None
	Record.setSublistValue(options)	None
	Record.setText(options)	None
	Record.setValue(options)	None
	Sublist.getColumn(options)	None
	record.attach(options) record.attach.promise(options)	10
	record.copy(options) record.copy.promise(options)	10 for transaction records 2 for custom records 5 for all other records
	record.create(options) record.create.promise(options)	10 for transaction records 2 for custom records 5 for all other records
	record.delete(options) record.delete.promise(options)	20 for transaction records 4 for custom records 10 for all other records
	record.detach(options) record.detach.promise(options)	10
	record.load(options) record.load.promise(options)	10 for transaction records 2 for custom records 5 for all other records
	record.submitFields(options) record.submitFields.promise(options)	10 for transaction records 2 for custom records 5 for all other records
	record.transform(options) record.transform.promise(options)	10 for transaction records 2 for custom records 5 for all other records
N/recordContext Module	recordContext.getContext(options)	10
N/redirect Module	redirect.redirect(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	redirect.toRecord(options)	None
	redirect.toRecordTransform(options)	None
	redirect.toSavedSearch(options)	5
	redirect.toSavedSearchResult(options)	5
	redirect.toSearch(options)	None
	redirect.toSearchResult(options)	None
	redirect.toSuitelet(options)	None
	redirect.toTaskLink(options)	None
N/render Module	TemplateRenderer.addCustomDataSource (options)	None
	TemplateRenderer.addQuery(options)	None
	TemplateRenderer.addRecord(options)	None
	TemplateRenderer.addSearchResults(options)	None
	TemplateRenderer.renderAsPdf()	None
	TemplateRenderer.renderPdfToResponse()	None
	TemplateRenderer.renderAsString()	None
	TemplateRenderer. renderToResponse(options)	None
	TemplateRenderer.setTemplateById(options)	None
	TemplateRenderer.setTemplateByScriptId (options)	None
	render.bom(options)	10
	render.create()	None
	render.mergeEmail(options)	None
	render.packingSlip(options)	10
	render.pickingTicket(options)	10
	render.statement(options)	10
	render.transaction(options)	10
	render.xmlToPdf(options)	10
N/runtime Module	Script.getParameter(options)	None
	Script.getRemainingUsage()	None
	Session.get(options)	None
	Session.set(options)	None
	User.getPermission(options)	None
	User.getPreference(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	runtime.getCurrentScript()	None
	runtime.getCurrentSession()	None
	runtime.getCurrentUser()	None
	runtime.isFeatureInEffect(options)	None
N/search Module (Search results are limited to 10,000 records; see Search Result Limits)	Column.setWhenOrderedBy(options)	None
	Page.next() Page.next.promise()	5
	Page.prev() Page.prev.promise()	5
	PagedData.fetch(options) PagedData.fetch.promise()	5
	Result.getValue(column) Result.getValue(options)	None
	Result.getText(column) Result.getText(options)	None
	ResultSet.each(callback) ResultSet.each.promise(callback)	10
	ResultSet.getRange(options) ResultSet.getRange.promise(options)	10
	Search.run()	None
	Search.runPaged(options) Search.runPaged.promise(options)	5
	Search.save() Search.save.promise()	5
	search.create(options) search.create.promise(options)	None
	search.createColumn(options)	None
	search.createFilter(options)	None
	search.createSetting(options)	None
	search.delete(options) search.delete.promise(options)	5
	search.duplicates(options) search.duplicates.promise(options)	10
	search.global(options) search.global.promise(options)	10

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	search.load(options) search.load.promise(options)	5
	search.lookupFields(options) search.lookupFields.promise(options)	1
N/sftp Module	Connection.download(options)	100
	Connection.list(options)	10
	Connection.makeDirectory(options)	10
	Connection.move(options)	10
	Connection.removeDirectory(options)	10
	Connection.removeFile(options)	10
	Connection.upload(options)	100
	sftp.createConnection(options)	None
N/sso Module	sso.generateSuiteSignOnToken(options)	20
N/suiteAppInfo Module	suiteAppInfo.isBundleInstalled(options)	5
	suiteAppInfo.isSuiteAppInstalled(options)	5
	suiteAppInfo.listBundlesContainingScripts (options)	10
	suiteAppInfo.listInstalledBundles()	10
	suiteAppInfo.listInstalledSuiteApps()	10
	suiteAppInfo.listSuiteAppsContainingScripts (options)	10
N/task Module	CsvImportTask.submit()	100
	EntityDeduplicationTask.submit()	100
	MapReduceScriptTask.submit()	20
	MapReduceScriptTaskStatus. getCurrentTotalSize()	25
	MapReduceScriptTaskStatus. getPendingMapCount()	10
	MapReduceScriptTaskStatus. getPendingMapSize()	25
	MapReduceScriptTaskStatus. getPendingOutputCount()	10
	MapReduceScriptTaskStatus. getPendingOutputSize()	25
	MapReduceScriptTaskStatus. getPendingReduceCount()	10
	MapReduceScriptTaskStatus. getPendingReduceSize()	25

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	MapReduceScriptTaskStatus.getPercentageCompleted()	10
	MapReduceScriptTaskStatus.getTotalMapCount()	10
	MapReduceScriptTaskStatus.getTotalOutputCount()	10
	MapReduceScriptTaskStatus.getTotalReduceCount()	10
	QueryTask.addInboundDependency(options)	None
	QueryTask.submit()	100
	RecordActionTask.submit()	50
	RecordActionTask.paramCallback	None
	ScheduledScriptTask.submit()	20
	SearchTask.addInboundDependency()	None
	SearchTask.submit()	100
	SuiteQLTask.addInboundDependency(options)	None
	SuiteQLTask.submit()	100
	WorkflowTriggerTask.submit()	20
	task.checkStatus(options)	None
	task.create(options)	None
N/task/accounting/recognition Module	MergeArrangementsTask.submit()	20
	MergeElementsTask.submit()	20
	recognition.checkStatus(options)	50
	recognition.create(options)	None
N/transaction Module	transaction.void(options) transaction.void.promise(options)	10
N/translation Module	translation.get(options)	1
	translation.load(options)	1
	translation.selectLocale(options)	None
N/ui/dialog Module	dialog.alert(options)	None
	dialog.confirm(options)	None
	dialog.create(options)	None
N/ui/message Module	Message.hide()	None
	Message.show()	None
	message.create(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
N/ui/serverWidget Module	Assistant.addField(options)	None
	Assistant.addFieldGroup(options)	None
	Assistant.addStep(options)	None
	Assistant.addSublist(options)	None
	Assistant.getField(options)	None
	Assistant.getFieldGroup(options)	None
	Assistant.getFieldGroupIds()	None
	Assistant.getFieldIds()	None
	Assistant.getFieldIdsByFieldGroup(fieldGroup)	None
	Assistant.getLastAction()	None
	Assistant.getLastStep()	None
	Assistant.getNextStep()	None
	Assistant.getStep(options)	None
	Assistant.getStepCount()	None
	Assistant.getSteps()	None
	Assistant.getSublist(options)	None
	Assistant.getSublistIds()	None
	Assistant.hasErrorHtml()	None
	Assistant.isFinished()	None
	Assistant.sendRedirect(options)	None
	Assistant.setSplash(options)	None
	Assistant.updateDefaultValues(values)	None
	AssistantStep.getFieldIds()	None
	AssistantStep.getLineCount(options)	None
	AssistantStep.getSublistFieldIds(options)	None
	AssistantStep.getSublistValue(options)	None
	AssistantStep.getSubmittedSublistIds()	None
	AssistantStep.getValue(options)	None
	Field.addSelectOption(options)	None
	Field.getSelectOptions(options)	None
	Field.setHelpText(options)	None
	Field.updateBreakType(options)	None
	Field.updateDisplaySize(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	Field.updateDisplayType(options)	None
	Field.updateLayoutType(options)	None
	Form.addButton(options)	None
	Form.addCredentialField(options)	None
	Form.addField(options)	None
	Form.addFieldGroup(options)	None
	Form.addPageInitMessage(options)	None
	Form.addPageLink(options)	None
	Form.addSecretKeyField(options)	None
	Form.addSublist(options)	None
	Form.addSubmitButton(options)	None
	Form.addSubtab(options)	None
	Form.addTab(options)	None
	Form.getButton(options)	None
	Form.getField(options)	None
	Form.getSublist(options)	None
	Form.getSubtab(options)	None
	Form.getTab(options)	None
	Form.getTabs()	None
	Form.insertField(options)	None
	Form.insertSublist(options)	None
	Form.insertSubtab(options)	None
	Form.insertTab(options)	None
	Form.removeButton(options)	None
	Form.updateDefaultValues(options)	None
	List.addButton(options)	None
	List.addColumn(options)	None
	List.addEditColumn(options)	None
	List.addPageLink(options)	None
	List.addRow(options)	None
	List.addRows(options)	None
	ListColumn.addParamToURL(options)	None
	ListColumn.setURL(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	Sublist.addButton(options)	None
	Sublist.addField(options)	None
	Sublist.addMarkAllButtons()	None
	Sublist.addRefreshButton()	None
	Sublist.getField(options)	None
	Sublist.getSublistValue(options)	None
	Sublist.setSublistValue(options)	None
	Sublist.updateTotallingFieldId(options)	None
	Sublist.updateUniqueFieldId(options)	None
	serverWidget.createAssistant(options)	None
	serverWidget.createForm(options)	None
	serverWidget.createList(options)	None
N/url Module	url.format(options)	None
	url.resolveDomain(options)	None
	url.resolveRecord(options)	None
	url.resolveScript(options)	None
	url.resolveTaskLink(options)	None
N/util Module	util.each(iterable, callback)	None
	util.extend(receiver, contributor)	None
	util.isAsyncFunction(obj)	None
	util.isArray(obj)	None
	util.isBoolean(obj)	None
	util.isDate(obj)	None
	util.isFunction(obj)	None
	util.isNumber(obj)	None
	util.isObject(obj)	None
	util.isRegExp(obj)	None
	util.isString(obj)	None
N/workbook Module	Workbook.runPivot(options)	10 per intersection returned
	workbook.create(options)	None
	workbook.createCalculatedMeasure(options)	None
	workbook.createColor(options)	None
	workbook.createConditionalFilter(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	workbook.createConditionalFormat(options)	None
	workbook.createConditionalFormatRule (options)	None
	workbook.createConstant(options)	None
	workbook.createDataDimension(options)	None
	workbook.createDataDimensionItem(options)	None
	workbook.createDataMeasure(options)	None
	workbook.createDimensionSelector(options)	None
	workbook.createDimensionSort(options)	None
	workbook.createExpression(options)	None
	workbook.createFieldContext(options)	None
	workbook.createFontSize(options)	None
	workbook.createLimitingFilter(options)	None
	workbook.createMeasure(options)	None
	workbook.createMeasureSelector(options)	None
	workbook.createMeasureSort(options)	None
	workbook.createMeasureValueSelector (options)	None
	workbook.createPathSelector(options)	None
	workbook.createPivotAxis(options)	None
	workbook.createPivot(options)	None
	workbook.createPositionPercent(options)	None
	workbook.createPositionUnits(options)	None
	workbook.createPositionValues(options)	None
	workbook.createReportStyle(options)	None
	workbook.createReportStyleRule(options)	None
	workbook.createSection(options)	None
	workbook.createSort(options)	None
	workbook.createSortByDataDimensionItem(options)	None
	workbook.createSortByMeasure(options)	None
	workbook.createSortDefinition(options)	None
	workbook.createStyle(options)	None
	workbook.createTable(options)	None
	workbook.createTableColumn(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	workbook.createTableColumnFilter(options)	None
	workbook.list()	10
	workbook.listPaged(options)	10
	workbook.load(options)	10
N/workflow Module	workflow.initiate(options)	20
	workflow.trigger(options)	20
N/xml Module	Document.adoptNode(options)	None
	Document.createAttribute(options)	None
	Document.createAttributeNS(options)	None
	Document.createCDATASection(options)	None
	Document.createComment(options)	None
	Document.createDocumentFragment()	None
	Document.createElement(options)	None
	Document.createElementNS(options)	None
	Document.createProcessingInstruction (options)	None
	Document.createTextNode(options)	None
	Document.getElementById(options)	None
	Document.getElementsByTagName(options)	None
	Document.getElementsByTagNameNS (options)	None
	Document.importNode(options)	None
	Element.getAttribute(options)	None
	Element.getAttributeNode(options)	None
	Element.getAttributeNodeNS(options)	None
	Element.getAttributeNS(options)	None
	Element.getElementsByTagName(options)	None
	Element.getElementsByTagNameNS(options)	None
	Element.hasAttribute(options)	None
	Element.hasAttributeNS(options)	None
	Element.removeAttribute(options)	None
	Element.removeAttributeNode(options)	None
	Element.removeAttributeNS(options)	None
	Element.setAttribute(options)	None

SuiteScript 2.x Module	API(s)	Governance Usage Units Used
	Element.setAttributeNode(options)	None
	Element.setAttributeNodeNS(options)	None
	Element.setAttributeNS(options)	None
	Node.appendChild(options)	None
	Node.cloneNode(options)	None
	Node.compareDocumentPosition(options)	None
	Node.hasAttributes()	None
	Node.hasChildNodes()	None
	Node.insertBefore(options)	None
	Node.isDefaultNamespace(options)	None
	Node.isEqualNode(options)	None
	Node.isSameNode(options)	None
	Node.lookupNamespaceURI(options)	None
	Node.lookupPrefix(options)	None
	Node.normalize()	None
	Node.removeChild(options)	None
	Node.replaceChild(options)	None
	Parser.fromString(options)	None
	Parser.toString(options)	None
	XPath.select(options)	None
	xml.escape(options)	None
	xml.validate(options)	None

API Governance Examples

The following examples show how governance units are calculated in a user event script and in a scheduled script:

Script Description	Usage Units Calculated
A user event script on a standard transaction record type (such as invoice) that includes: ■ one call to record.delete(options) ■ one call to Record.save(options)	This script uses a total of 40 usage units: ■ record.delete(options) uses 20 usage units for a transaction record ■ Record.save(options) uses 20 usage units for a transaction record Each user event script can use a maximum of 1,000 usage units, so in this case, this script has plenty of room to be expanded.

Script Description	Usage Units Calculated
A scheduled script on a standard non-transaction record type (such as customer) that includes: ■ one call to record.load(options) ■ one call to record.transform(options) ■ one call to email.send(options)	This script uses a total of 30 usage units: ■ record.load(options) uses 5 usage units for a standard non-transaction record ■ record.transform(options) uses 5 usage units for a non-standard transaction record ■ email.send(options) uses 20 usage units Each scheduled script can use a maximum of 10,000 usage units, so in this case, this script has plenty of room to be expanded.

For more examples, see Monitoring Script Usage.

Monitoring Script Usage

Applies to: SuiteScript 2.x | SuiteCloud Developer

You can monitor SuiteScript unit usage using the Script.getRemainingUsage() method as shown in the following examples.

Example 1

This example shows how to instantiate the current script object and call Script.getRemainingUsage() to write the script's remaining usage units to the execution log.

```
var scriptObj = runtime.getCurrentScript();
log.debug({
    title: "Remaining usage units: ",
    details: scriptObj.getRemainingUsage()
});
```

Example 2

This example shows how to instantiate the current script object and check for the number of remaining usage units. If there are more than 50 usage units remaining, the script will execute a certain set of instructions.

```
var scriptObj = runtime.getCurrentScript();
var unitsRemaining = scriptObj.getRemainingUsage();
if (unitsRemaining > 50)
{
    //execute code here
}
```

You can also monitor script unit usage by running the script in the SuiteScript Debugger. After a script completes execution, unit usage details are displayed on the Execution Log tab in the SuiteScript Debugger console. If usage units are exceeded, usage limit error messages are emailed to the script owner indicating the number of usage units that were executed in the script before the usage limit error occurred.

For more information about the SuiteScript Debugger, see the help topic SuiteScript Debugger.

Governance on Script Logging

Applies to: SuiteScript 2.x | SuiteCloud Developer

NetSuite governs the amount of script execution logging that can be done. The governance model is as follows:

- Within a 60–minute time period, a company is allowed to make up to 100,000 N/log module method calls across all of their scripts.
- System error logs are purged after 60 days, and user-generated logs are purged after 30 days.

Use the N/log Module to access methods for logging script execution details.

Script owners are notified if NetSuite detects that a script is logging excessively and NetSuite automatically adjusts the log level. This ensures that the offending script continues to execute and helps prevent an inordinate amount of logging from one company. For example, Spruce Street Foods has 10 scripts running during a 60–minute time period. If one of the scripts calls log.debug(options) 70,000 times within each 20–minute time period, NetSuite will automatically raise the script's log level.

The change to the log level will appear in the Log Level field on the script's Script Deployment page:

If the offending script's log level was originally set to Debug, NetSuite will increase the level to Audit. This means that the `log.debug` line of code will continue to execute, however, nothing will be logged, because the log level for the script has been raised to Audit and the line of code is using the Debug level. For more information about log levels, see the help topic Using Log Levels. For more information about viewing script execution logs, see the help topic Script Execution Logs.

For more information, see the help topic N/log Module Guidelines.

Script Execution Logs

All script execution logs in NetSuite are stored for a maximum of 30 days. You can access script execution logs using the Execution Log tab, using a Server Script Log search, or using the Script Execution log page. Details are shown in the following table:

Access to Script Execution Logs	Details
▪ Execution Log tab on script and script deployment records	The capacity for script execution logs on the Execution Log tab is shared by customers on the same NetSuite database. For further protection against excessive logging, script execution logs are governed by a total storage limit of 5 million logs on each instance of the database. If this limit is reached, all logs across all customers on that database are purged. If you require the ability to view all logs up to 30 days on the Execution Log tab, consider creating a custom solution using custom records to store log information from logs.

Access to Script Execution Logs	Details
	For more information, see the help topic Using the Script Execution Log Tab.
■ Server Script Log search	The capacity for script execution logs returned by the Server Script Log search is shared by all customers on the same NetSuite database. For further protection against excessive logging, script execution logs are governed by a total storage limit of 5 million logs on each instance of the database for this search. On each NetSuite server, if this limit is reached, logs across all customers on that server are purged. If you require the ability to search all logs up to 30 days, consider creating a custom solution using custom records to store log information from logs. If a server script log search is not returning logs less than 30 days old, go to Customization > Scripting > Script Execution Logs and use the filters at the top of the Script Execution Log page to search for logs. You may also want to check script deployments that have large amounts of logging and consider lowering the Log Level to Error or Emergency or limiting the number of log lines in your script.
■ Script Execution log page	The execution logs shown on the Script Execution log page are stored for 30 days regardless of volume. For more information about the Script Execution log, see the help topic Viewing a List of Script Execution Logs.

Script Owner Notifications

If NetSuite detects that a script is logging excessively, the owner of the script is notified that the script is logging excessively and could possibly exceed the 100,000 logging threshold (for a 60–minute time period).

NetSuite sends email notifications and adds a log entry to the script's Execution Log indicating that a script's log level has been increased. For more information about script execution log levels, see the help topic Setting Script Execution Log Levels.

Search Result Limits

Applies to: SuiteScript 2.x | SuiteCloud Developer

Search results are limited to 1,000 records when you execute SuiteScript searches using the N/search Module. For information about working with SuiteScript searches in NetSuite, see the help topic N/search Module.

Note that if you load an existing saved search using search.load(options), and then call Search.run() to return a result set of search.ResultSet objects, you may get up to 4,000 results returned. For more information, see the help topic ResultSet.each(callback).

Script Execution Time Limits

Applies to: SuiteScript 2.x | SuiteCloud Developer

Each server script type and plug-in type has a limit on the amount of time it can run in a single execution. If the time limit is exceeded, an `SSS_TIME_LIMIT_EXCEEDED` error is thrown and script execution stops.

However, a script may run for a long time, and you may not see the SSS_TIME_LIMIT_EXCEEDED error under these conditions:

- The script performs a large number of record operations without exceeding the usage unit limit. See Script Type Usage Unit Limits for usage unit limits.
- The script causes a large number of user event scripts or workflows to execute.
- The script performs database queries or updates that collectively take a long time to finish.

The following tables list the time limit (in seconds) for each script type, core plug-in type, and custom plug-in type.

Script Type	Time Limit (in Seconds)
Bundle Installation Scripts (SuiteScript 1.0) SuiteScript 2.x Bundle Installation Script Type	3,600
Client Scripts (SuiteScript 1.0) SuiteScript 2.x Client Script Type	300
Custom record action	300
Map/reduce (input stage) SuiteScript 2.x Map/Reduce Script Type	3,600
Map/reduce (map stage) SuiteScript 2.x Map/Reduce Script Type	300
Map/reduce (reduce stage) SuiteScript 2.x Map/Reduce Script Type	900
Map/reduce (summarize stage) SuiteScript 2.x Map/Reduce Script Type	3,600
Mass Update Scripts (SuiteScript 1.0) SuiteScript 2.x Mass Update Script Type	300
Portlet Scripts (SuiteScript 1.0) SuiteScript 2.x Portlet Script Type	300
RESTlets (SuiteScript 1.0) SuiteScript 2.x RESTlet Script Type	300
SuiteScript 2.x SDF Installation Script Type	3,600
Single-page application (SPA)	300
Scheduled Scripts (SuiteScript 1.0) SuiteScript 2.x Scheduled Script Type	3,600
Suitelets (SuiteScript 1.0) SuiteScript 2.x Suitelet Script Type	300
SuiteScript Server Pages (SSP) application	300
User Event Scripts (SuiteScript 1.0)	300

Script Type	Time Limit (in Seconds)
SuiteScript 2.x User Event Script Type	
Workflow Action Scripts (SuiteScript 1.0) SuiteScript 2.x Workflow Action Script Type	300

Core Plug-in Type	Time Limit (in Seconds)
Adjusted Consolidated Rates	300
Bank Connectivity Plug-in	300
Bank Statement Parser Plug-in	1,800
Custom GL Lines Plug-in	30
Dataset Builder Plug-in	300
Email Capture Plug-in	300
Financial Institution Connectivity Plug-in	3,600
Financial Institution Parser Plug-in	1,800
Payment Processing	300
Platform Extension	300
Promotions	300
Revenue Management	300
Shipping Partners	300
Tax Engine	60
Workbook Builder Plug-in	300

Custom Plug-in Type	Time Limit (in Seconds)
Plug-in Type	3,600
Plug-in Type Implementation	3,600

SuiteScript Best Practices

Applies to: SuiteScript 2.x | SuiteCloud Developer

General Development Best Practices

Applies to: SuiteScript 2.x | SuiteCloud Developer

Important: If you are using SuiteScript 1.0 or SuiteScript 2.0 for your scripts, consider converting these scripts to SuiteScript 2.1. Use SuiteScript 2.1 to take advantage of new features, APIs, and functionality enhancements, including language features that are supported in the ECMAScript 2019 specification. For more information, see the help topics SuiteScript Versioning Guidelines and SuiteScript 2.1.

The following are general best practices for writing SuiteScript scripts:

General	■ Write your SuiteScript code, not in a clever manner, but rather in a straightforward, obvious way. ■ Follow the Do Not Repeat Yourself (DNRY) principle. Consider creating reusable functions.
Code organization and structure	■ Good code is well organized. Data and operations in each function or class fit together. Organize your code into reusable chunks. Many functions can be used in a variety of forms. Any reusable functions should be stored in a common library file and then called into specific event functions for the required forms as needed. ■ Good code is written using small, readable, reusable functions. Including too many lines of code in one function makes it harder to understand and debug. Consider refactoring your code to keep all functions reasonable in size. ■ Use nesting judiciously. Extensive nesting may make your script unreadable. Try to keep your scripts as readable as possible. ■ During script development, componetize your scripts, load them individually and then test each one -- inactivating all but the one you are testing when multiple components are tied to a single user event.
Naming conventions	Good code uses meaningful naming conventions. A well written script containing descriptive names for functions, variables, IDs, provides a good structure for building comments around. Functions: ■ Although you can use any desired naming conventions for functions within your code, you should use custom name spaces or unique prefixes for all your function names. ■ When working with the top-level NetSuite functions in Client SuiteScript (for example the pageInit function), you should name the new function to correspond to the NetSuite name. For example, a pageInit function can be named pageInit or formAPageInit. If your code

	is already established, you can wrap it with a top-level function that has the appropriate naming convention. ■ Function Names: All function names should be written in lower camel case which means the first letter of each word (except the first) is capitalized. ■ For IDs, enter all record, field, sublist, tab, and subtab IDs in lower case in your SuiteScript code. Prefix all custom script IDs and deployment IDs with an underscore (_). Variables and Constants: ■ Avoid using a single character as a variable name. ■ Name your variables to identify the data type and describe the data stored in the variable. For example, □ String variables – prefix with "st", such as stTitle □ Integer variables – prefix with "int", such as intTotalCount □ Float variables – prefix with "fl", such as flPrice □ Boolean variables – prefix with "b", such as bIsDone □ Array variables – prefix with "arr", such as arrPhoneCalls □ Object variables – prefix with "Obj", such as ObjNewPet □ Record variables – prefix with "rec", such as recCustomer □ Date variables – prefix with "dt", such as dtFirstBillingDate ■ Local variable scope is only within the function. Use standard camelCase naming. ■ Global variable scope is within the whole file or more files if declared under included library. Use a variable in upper case spaced by underscore. ■ To represent a pseudo constant, use a variable in upper case spaced by underscore. Constants are to be used to make code more readable. IDs: ■ For IDs, enter all record, field, sublist, tab, and subtab IDs in lower case. Prefix all custom script IDs and deployment IDs with an underscore (_). File names: ■ If you write SuiteScript code for multiple accounts, consider using the following file name convention: <Company Name/Abbreviation>_<Script Type>_<Requirement Description>.js. For example, MyCompany_CS_SetTaxable.js. The following are suggested Script Types: □ CS – client scripts □ UE – user events □ SL – Suitelet □ RL – RESTlet □ PL – Portlet □ SC – Scheduled □ MR – Map/Reduce □ GI – SuiteGL □ WA – Workflow Action □ MU – Mass Update □ BI – Bundle Installation
Commenting	■ Thoroughly comment your code. Include comments about what the script is doing and who authored it. This practice helps with debugging and development and assists NetSuite Customer Support in locating problems, if necessary. It also helps anyone else who works with the script.

	■ Write useful comments. Good code comments explain what is being done and why it is being done. ■ When documenting functions try to provide a good abstract of the actions of the function without restating the function name. ■ A good way to write relevant comments is to plan your scripts using pseudo code and replace the pseudo code with a relevant comment where necessary. ■ At a minimum, you should include comments: (1) for the file level disclaimer, (2) for version control, (3) at the function level, and (4) at the logic level.
White space and formatting	■ Use white space and formatting to enhance readability rather than to minimize typing effort. The following rules should always apply: □ All line returns within a function should be terminated with a ; (semicolon). □ Code blocks should be tab indented.
Using IDs	■ Use **static** ID values in your API calls where applicable because name values can change.
Passwords	■ Do not hard-code any passwords in scripts. The password and password2 fields are supported for scripting. For additional information, see SuiteScript Security Considerations.
Error and exception handling	■ Include proper error handling sequences in your script wherever data may be inconsistent, not available, or invalid for certain functions. For example, if your script requires a field value to validate another, ensure that the field value is available. ■ Use try-catch blocks. Try-catch blocks provide a way to "catch" an exception object thrown with the try block. As soon as an exception is thrown the catch block starts. An optional finally clause can also be used to run (guaranteed) if any exception occurs or not.
URLs	■ Do not hard-code URL links. If the domain name changes, hard-coded URLs will not work. To prevent issues with links, use the N/url Samples.
Date and Currency fields	■ Use the built-in library functions whenever possible for reading/writing Date/Currency fields and for querying XML documents.

Script parameters	■ Use script parameters or configuration files wherever possible, instead of hard coding values, to ensure that scripts can be configured easily
	■ You can use the runtime.getCurrentScript() function in the runtime module to reference script parameters. For example, use the following code to obtain the value of a script parameter named custscript_case_field:
	`define(['N/runtime'], function(runtime) {`
	`  function pageInit(context) {`
	`  var strField = runtime.getCurrentScript().getParameter('SCRIPT', 'custscript_case_field');`
	`  ...`
	`  });`
	■ For security reasons, do not include confidential information in script parameters. Information saved in script parameters can be indexed by search engines and therefore be viewable by the public. This means, for example, that the information could be found in Google searches. For more information, see Creating Script Parameters Overview.

Reserved words	■ Ensure that your scripts do not use any reserved words in SuiteScript. For more information, see the help topic SuiteScript Reserved Words.
Execution contexts	■ Consider the most appropriate script execution context for your script. The script execution context specifies how and when a script is triggered to execute. If you don't use the appropriate context, blocks of scripts run when they shouldn't and result in unexpected errors in the data and the script. For more information, see the help topic Execution Contexts.

Loading records	■ For record Create or Edit operations, you should load records in dynamic mode. This is required to make updates on record object. Load records in a non-dynamic mode if you want to only get field values from the record and update on record is note needed. ■ Do not use record.load(options) to load a whole record when you only need to get the value of a few fields. The performance of the load record API is similar to loading the record in the user interface without field rendering since it loads all configuration fields and sublists for the record. To avoid slowing down the execution of the script, use search.lookupFields(options) to get the value of a few fields.
Invoking methods	■ You should use parameter id's instead of parameter sequence. ■ You should use defined SuiteScript 2.0 native enums. You may find the list of valid enums on Help and SuiteScript 2.0 API pdf.
Asynchronous programming	■ When using promises in your SuiteScript scripts to implement asynchronous programming, you should consider the best practices listed in the Best Practices for Asynchronous Programming with SuiteScript help topic.
Custom code	■ Place all custom code and markup, including third party libraries, in your own namespace. **Important:** Custom code must not be used to access the NetSuite DOM. Developers must use SuiteScript APIs to access NetSuite UI components.
Testing	■ Good code is well-tested. Testing serve as an executable specification of the code and examples of its use. ■ Always thoroughly test your code before using it on your live NetSuite data. ■ If the same script is used across multiple forms, ensure that you test any changes to the code for **each** form that the code is associated with. ■ During testing, the Deployment Status should be Testing to limit the script execution to only the script owner. If several users are testing a script (or if a Sandbox Account is being used), the Deployment Status can be set to Released, but you must be sure to choose the appropriate Audience so that the script does not execute for users that are not involved in the testing process. For production, the Deployment Status should set be Released. Again, it is critical to set the appropriate Audience so that the script does not execute for unintended users. ■ During testing, the Log Level should be set to Debug.
Logging	■ Log messages in the script should make business sense; include the record id, record type and other important details in your log messages. Each log message should provide a clear picture of the functionality that the script is designed to achieve. ■ Key steps in each script should be logged as Audit. ■ Proper logging adds to the readability of the code and also significant on issue/error debugging. ■ During testing, the Log Level should be set to Debug. For a Production Deployment, the Log Level should be set to Audit. Note that if there are production issues that require diagnosis/ debugging, the Deployment Log Level can be set back to Debug.
Other	■ Consider the time zone that your scripts must use. By default, server scripts use the time zone that is specified in your NetSuite account. Be sure to convert your time values to the correct time zone before you use them, if necessary. ■ Consider field limits and translation when creating your scripts. Fields that use special characters used in some languages may support fewer characters than the field limit specified. For example, the Reference No. field (tranid) supports 45 English characters, but supports fewer Chinese characters.

	■ Be sure to use BigInt values in your scripts to avoid potential errors. Integers larger than MAX_SAFE_INTEGER and smaller than MIN_SAFE_INTEGER must be enclosed in quotations or appended with '*n*' at the end of the integer literal. For more information about BigInt values, see BigInt.

You can also refer to the following Internet resources for JavaScript coding best practices:

- JavaScript Best Practices (w3)
- JavaScript Best Practices (w3 schools)

Client Script Best Practices

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following are best practices for both form-level and record-level client SuiteScript development.

General	■ Use global (record-level) client scripts to get a more flexible deployment model and to port (include in a bundle or SuiteCloud project) than client scripts attached to forms.
Using SuiteScript modules	■ When you specify the modules that your script uses, make sure that the order of the modules matches the order of the parameters in your main function. For example, if your script uses the N/error, N/record, and N/file modules and you specify them in this order, make sure that your function parameters are in the same order, as follows: `require(['N/error', 'N/record', 'N/file'], function(error, record, file) { ... });`
Using Record.setValue and Record.setCurrentSublistValue methods	■ Record.setValue(options) and Record.setCurrentSublistValue(options) execution is multi-threaded whenever child field values need to be sourced in. To synchronize your logic, use the `postSourcing` function or set the `forceSyncSourcing` synchronous parameter to `true`.
Advanced Employee Permissions	■ When the Advanced Employee Permissions feature is enabled, the search columns available to users is also dependent on the permissions assigned to the role. ■ When the Advanced Employee Permissions feature is enabled, the fields and sublists a user has access to can change depending on which employee is being viewed or edited. This is different from other records in NetSuite, where permissions granted to a role determines what the role can see for every instance of that record. In general, scripts should always verify that a field exists before trying to do something with it. Simply calling functions and methods that interact with fields before checking whether the field is there may result in inconsistent behavior. For example, the department field is permitted on the employee record. When you verify that the field exists and you do not have access, a null value is returned, and if the field is empty, an empty string is returned. □ Unsafe Practice Example `var employeeRecord = record.load ({ type: record.Type.EMPLOYEE, id: 115 }); employeeRecord.setValue({ fieldId: 'department', value: 'marketing' });`

	`employeeRecord.save();`
	□ Safe Practice Example
	To detect if your role has access to a field for a specific employee, load the employee record object and call `getFields()`. If the field exists and you do have access a `true` value is returned. In the below example, the user has access to the department field for the employee with ID: 115.
	`var employeeRecord = record.load ({ type: record.Type.EMPLOYEE, id: 115 }); var accessToDepartment = employeeRecord.getFields().includes('department');`

Editing sublists	■ When you are editing the current line of a sublist and a nested client script is triggered (for example, when a `fieldChanged` script is triggered by changing a value in the current sublist line), make sure that the nested script does not select a different line of the same machine as the current line. Otherwise, the changes made in the parent script may be lost.
Execution contexts	■ Use execution context filtering to specify how and when a client script is executed. Execution contexts provide information about how a script is triggered to execute. For example, a script can be triggered in response to an action in the NetSuite application, or an action occurring in another context, such as a web services integration. You can use execution context filtering to ensure that your scripts are triggered only when necessary. For more information, see the help topic Execution Contexts.
setValue and setText methods	■ Methods that set values accept raw data of a specific type and do not require formatting or parsing. Methods that set text accept strings but can conform to a user-specified format. For example, when setting a numerical field type, `setValue()` accepts any number, and `setText()` accepts a string in a specified format, such as “2,000.39” or “2.00,39”. When setting a date field type, `setValue()` accepts any `Date` object, and `setText()` accepts a string in a specified format, such as “6/9/2016” or “9/6/2016”.
Debugging	■ When debugging client scripts, you can insert a `debugger;` statement in your script, and execution will stop when this statement is reached: `function runMe() { var k = 1; k *= (k + 9); debugger; console.log(k); }` When your script stops at the `debugger;` statement, you can examine your script properties and variables using the debugging tools in your browser. ■ When debugging client scripts, some scripts might be minified. Minified scripts have all unnecessary characters removed, including white space characters, new line characters, and so on. Minifying scripts reduces the amount of data that needs to be processed and reduces file size, but it can also make scripts difficult to read. You can use your browser to de-minify scripts so that they’re more readable. To learn how to de-minify scripts, see the documentation for your browser.
Testing	■ When testing form-level client scripts, use Ctrl-Refresh to clear your browser cache and ensure that the latest scripts are being executed.

Map/Reduce Script Best Practices

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following are best practices for working with map/reduce scripts.

General	As described in Hard Limits on Function Invocations, NetSuite imposes governance limits on single invocations of map, reduce, getInputData, and summarize functions: ■ **map**: 1,000 usage units (the same as mass update scripts) ■ **reduce**: 5,000 usage units ■ **getInputData**: 10,000 usage units ■ **summarize**: 10,000 usage units If you are concerned about potential issues with these limits, review your script to make sure that your map and reduce functions are relatively lightweight. Your map and reduce functions should not include a long or complex series of actions. For example, consider a situation in which your map or reduce function loads and saves multiple records all at the same time. This approach might cause an issue with the limits described above. If your getInputData function returns a list of record IDs, a better approach might be to use the map function to load each record, update fields on the record, and save it. If you have a script that performs a significantly more complex series of operations within a single function (such as loading and saving multiple records, or transforming multiple records), consider using a different script type, such as a scheduled script.

Passing search data to getInputData	In the getInputData stage, your script must return an object that can be transformed into a list of key/value pairs. A common approach is to use a search. If you decide to use this technique, note that you should have your function return either a search.Search object or an object reference to a saved search. By contrast, if you execute a search within the getInputData function and return the results (for example, as an array), there is a greater risk that the search will time out.
	Instead, you should use one of the following approaches:
	■ Return a search object. That is, return an object created using search.create(options) or search.load(options).
	■ Return a search object reference. That is, return an inputContext.ObjectRef object that references a saved search.
	In both cases, the time limit available to the search is more generous than it would be for a search executed within the function.
	The following snippet shows how to return a search object:
	`function getInputData()`
	`{`
	`  return search.create({`
	`  type: record.Type.INVOICE,`
	`  filters: [['status', search.Operator.IS, 'open']],`
	`  columns: ['entity'],`
	`  title: 'Open Invoice Search'`
	`  });`
	`}`

	And the following snippet shows how to return a search object reference:
	
	`...`
	`function getInputData {`
	`{`
	`// Reference a saved search with internal ID 1234.`
	
	`return {`
	`type: 'search',`
	`id: 1234`
	`};`
	`}`
	`...`
	
	For information about additional ways to return data, see the help topic getInputData(inputContext).

Minimizing risk of data duplication	A map/reduce script can be interrupted at any time by an application server disruption. Afterward, the script is restarted. Depending on how the script is configured, when a map or reduce job starts again, it may attempt to retry processing for the same key/value pairs it had flagged for processing when the interruption occurred. Similarly, if an uncaught error disrupts the job, the system may retry processing for the pair that was being processed when the error occurred. **Note:** For an overview of the system's behavior following an interruption, see the help topic System Response After a Map/Reduce Interruption. When a job is restarted, there is an inherent risk of data duplication. To minimize this risk, use the following guidance.
Handling restarts	Every map/reduce script should be written in such a way that each entry point function checks to see whether the function has been previously invoked. To do this, use the context.isRestarted property, which exists for every map/reduce entry point. If the function has been restarted, the script should provide any logic needed to avoid duplicate processing. For examples, see the help topic Adding Logic to Handle Map/Reduce Restarts.
Buffer size	When you deploy a script, the deployment record includes a field called Buffer Size. The default value of this field is 1. In general, you should leave this value set to the default. The purpose of the Buffer Size field is to control how many key/value pairs are flagged for processing at one time, and how frequently a map or reduce job saves data about its progress. Setting this field to a higher value may have a small performance advantage. However, the disadvantage is that, if the job is interrupted by an application server restart, there is a greater likelihood of one or more key/value pairs being processed twice. For that reason, you should leave this value set to 1, particularly if the script is processing records. For more details on this field, see the help topic Buffer Size.

Scheduled Script Best Practices

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following are best practices for scheduled scripts. Also see the help topic Scheduled Script Handling of Server Restarts.

General	■ You should set your scheduled scripts to run during the hours of 2 AM to 6 AM PST. Scripts set to run during the hours of 6 AM to 6 PM PST may not run as quickly due to high database activity. ■ The number of **Not Scheduled** deployments to create should depend on the anticipated number of simultaneous calls you expect to make to this script and the approximate execution time of the script. A general rule of thumb is to create twice as many deployments as the total number of simultaneous calls you anticipate for this script. ■ **Scheduled** deployments and **Not Scheduled** deployments are executed from the same processor pool. Keep this in mind as you deploy multiple scheduled scripts because the amount pending script instances is the ultimate bottle-neck for scheduled script execution throughput. ■ Although there is no restriction on the number of **Not Scheduled** scripts that can be placed into the processing pool, too many pending scripts may create a backlog and compromise system performance. Because only one script can be run in a processor at a time, you should not overload the system. ■ If you want to deploy scheduled scripts that are **scheduled to run hourly on a 24 hour basis**, the following sample values should be set on the Script Deployment page: □ Deployed = checked □ Daily Event = [radio button enabled] □ Repeat every 1 day □ Start Date = [today's date] □ **Start Time = 12:00 am** □ Repeat = every hour □ End By = [blank] □ No End Date = checked □ Status = Scheduled □ Log Level = Error □ Execute as Role = Set to **Administrator** If the **Start Time** is set to any other time than 12:00 am (for example it is set to 2:00 pm), the script will start at 2:00 pm, but then finish its hourly execution at 12:00 am. It will not resume until the next day at 2:00 pm. ■ When possible, schedule your deployments during the hours of 2 AM to 6 AM PST. Deployments scheduled for submission during the hours of 6 AM to 6 PM PST may not run as quickly due to high database activity. ■ The number of **Not Scheduled** deployments to create should depend on the anticipated number of simultaneous calls you expect to make to this script and the approximate execution time of the script. A general rule of thumb is to create twice as many deployments as the total number of simultaneous calls you anticipate for this script.
Deployments that continue to use queues	■ If you do not have an account with SuiteCloud Plus, all **Scheduled** deployments and **Not Scheduled** deployments that continue to use queues are executed from the same queue. If you deploy multiple scheduled scripts that continue to use queues, your queue size is the ultimate bottle-neck for scheduled script execution throughput. For additional information, see the help topic Scheduled Script Deployments that Continue to Use Queues.

	■ Although there is no restriction on the number of **Not Scheduled** scripts that can be submitted to SuiteCloud Processors, too many waiting scripts may create a backlog and compromise system performance. Because only one script can be run at a time, you should not overload the system. ■ Although there is no restriction on the number of **Not Scheduled** scripts that can be placed into the processing pool, too many pending scripts may create a backlog and compromise system performance. Because only one script can be run in a processor at a time, you should not overload the system.

Suitelets and UI Object Best Practices

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following are best practices for Suitelet development using UI objects and custom UI.

General	■ Suitelets are ideal for generating NetSuite pages (forms, lists), returning data (XML, text), and redirecting requests. ■ Limit the number of UI objects on a page (< 100 rows for sublists, < 100 options for on demand select fields, < 200 rows for lists).
HTML	■ Experiment with inline HTML fields embedded on the form before going the full custom HTML page route.
iFrames	■ Append "ifrmcntnr=T" to the external URL when embedding in iFrame especially if you are using Firefox. (For more about NetSuite and iFrame, see the help topic Embedding an Online Form in your Website Page.)
User credentials	■ When building custom UI outside of the standard NetSuite UI (such as building a custom mobile page using Suitelet), use the User Credentials APIs to help users manage their credentials within the custom UI. For more information, see User Credentials APIs. ■ When building custom UI outside of the standard NetSuite UI (such as building a custom mobile page using Suitelet), use the User Credentials APIs to help users manage their credentials within the custom UI.
Calling a Suitelet and redirection	■ When calling a Suitelet using its external URL, properly escape the parameter values to avoid cross-site scripting injections, for example, by converting the appropriate characters to HTML entities. ■ For access or redirection from another script to a Suitelet, the best practice is to use url.resolveDomain(options) to discover the URL instead of hard- coding the URL.
Advanced Employee Permissions	■ When the Advanced Employee Permissions feature is enabled keep the following in mind: □ To avoid inadvertently exposing employee data, use caution when running Suitelets or Restlets as an administrator. A user with a role that has limited access to the employee record can access a Suitelet or Restlet that runs as an administrator. Depending on how the Suitelet or Restlet is written, the user may have access to employee information that they would otherwise not see. □ Use caution when setting up Suitelets and Restlets to give access to users without having to log in since it could potentially expose employee information in uncontrolled ways.
Deployment	■ Deploy Suitelets as "Available without Login" only if necessary (no user context, login performance overhead). (See Setting Available Without Login.)

User Event Script Best Practices

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following are best practices for developing user event scripts.

General	■ Mission-critical business logic implemented using user events should be accompanied by a clean–up scheduled script to account for any unexpected errors or misfires. ■ Do not try to execute a user event script from another user event script. Instead, create a module containing the code that is common between the two user event scripts, and use the module in both scripts. ■ Make sure that the user event script does not read a sensitive field value.
Context	■ Use the `type` argument and `context` object to define and limit the scope of your user event logic. See the help topic context.UserEventType. ■ Use the `context` argument and `context.UserEventType` enum to define and limit the scope of your user event logic. See the help topic context.UserEventType.
Entry points	■ Any operation that depends on the submitted record being committed to the database should happen in an `afterSubmit` script. ■ When updating transaction line items in a `beforeSubmit` script, ensure that the line item totals net taxes and discounts are equal to the `summarytotal`, `discounttotal`, `shippingtotal`, and `taxtotal` amounts. ■ Assigning many executable functions to one record type is discouraged because this could negatively affect the user experience with that record type. For example, if there are ten `beforeLoad` scripts that must complete their execution before the record loads into the browser, the time needed to load the record may increase significantly. Be aware of the number of user events scripts used, including bundled user event scripts. ■ To set a field on a record or make any changes to a record being submitted, use the `beforeSubmit` event. ■ Perform all post-processing operations of the current record on an `afterSubmit` event.
Storing values	■ If you want to store a value during a `beforeLoad` operation and then read that value during an `afterSubmit` operation in the same script, consider using a hidden custom field to store the value. You can add a hidden custom field to the form and store your value during the `beforeLoad` operation, and you can retrieve the value from the same field during the `afterSubmit` operation.
Execution contexts	■ Use execution context filtering to specify how and when a user event script is executed. Execution contexts provide information about how a script is triggered to execute. For example, a script can be triggered in response to an action in the NetSuite application, or an action occurring in another context, such as a web services integration. You can use execution context filtering to ensure that your scripts are triggered only when necessary. For more information, see the help topic Execution Contexts.
Performance	■ Limit the amount of script execution in user event scripts (less than five seconds, if possible) since they run often and in-line. You can use the Application Performance Management (APM) SuiteApp to test the performance of your scripts deployed on a specific record type. See the help topic Application Performance Management (APM).
Debugging	■ When your script stops at the `debugger;` statement, you can examine your script properties and variables using the debugging tools in your browser. You can also use the `debugger;` statement in the SuiteScript Debugger to help you debug server scripts. For more information about the SuiteScript Debugger, see SuiteScript Debugger.

Hosted websites	■ Activities (user events) on a hosted website can trigger server SuiteScripts. In addition to sales orders, scripts on case records and customer records also execute in response to web activities.

Optimizing SuiteScript Performance

Applies to: SuiteScript 2.x | SuiteCloud Developer

There are certain changes you can make to your scripts to ensure the execute with performance in mind. This may be particularly true for custom scripts. You can see if there are custom scripts in your account at Customization > Scripting > Scripts. The following guidelines are suggested to optimize script performance:

- General Scripting Guidelines
- Accessing Records in Scripts
- Scripting Searches

General Scripting Guidelines

- Save retrieved values on a variable and use again on succeeding calls.
- Execute a block of code only if it satisfies a condition.
- Remove redundant operations or actions and lines of code which are no longer used within the script execution.
- Remove duplicate logic on a script that is already performed by other events.
- Combine similar scripts/functions whenever possible.
- Inactivate scripts/deployments that are no longer needed.
- Use asynchronous processing for user events.
- Use built-in permissions (by leveraging Run as Role) instead of scripting permissions.
- Consider using SuiteScript 2.1 language constructs and converting your SuiteScript 2.0 scripts to SuiteScript 2.1. Some SuiteScript 2.1 language constructs can be used to improve script performance. For more information about SuiteScript 2.1, see the help topic SuiteScript 2.1.
- As a general rule, design your user event scripts to execute in under 5 seconds, your Suitelets and Portlets to execute in under 10 seconds, and your scheduled scripts in under 5 minutes. This gives you a large enough margin of error to handle the outlier use cases (where the volume of work is unusually large, or the overall system is slow due to high load).
- To minimize execution logging after your script is tested and released, set your script log level to ERROR or EMERGENCY. See Setting Script Execution Log Levels.
- Deploy scripts to run as administrator only if necessary to minimize security risk and to eliminate performance overhead. See Executing Scripts Using a Specific Role.
- Use execution context filtering to specify how and when a client script or user event script is executed. Execution contexts provide information about how a script is triggered to execute. For example, a script can be triggered in response to an action in the NetSuite application, or an action occurring in another context, such as a web services integration. You can use execution context filtering to ensure that your scripts are triggered only when necessary. For more information, see the help topic Execution Contexts.

Accessing Records in Scripts

- If applicable, change the trigger of a script to avoid reloading a record.
- Avoid saving records multiple times in an After Record Submit event.
- Avoid loading and submitting a record on a Before Record Submit trigger.
- Minimize API calls that perform load, search, or save record operations.
- Use inline editable child custom records whenever your use case calls for batch processing of multiple related/child records during user events on the parent record. (See the help topic Custom Child Record Sublists in the NetSuite Help Center.)

Scripting Searches

- Optimize search filters and columns. Make your search results faster by:
  - Filtering inactive records.
  - Entering shorter date range criteria.
  - Using faster operators such as starts with/between/within instead of contains/formulas.
  - Removing unnecessary columns.
- Remove search results columns that are not used by a script. Place any lines that add columns to search results in a comment. Then the returned values are not used by the succeeding script logic.
- Combine searches of the same record type by using merged filters. Wherever possible, combine searches for the same record into one main search with merged filters. This improves performance by minimizing search instances.

SuiteScript Security Considerations

Applies to: SuiteScript 2.x | SuiteCloud Developer

Certain measures must be taken to ensure your SuiteScript script executes in a safe, secure manner. This includes the use of user credentials and executing scripts using a specific role, particularly when working with sensitive data. In general, you should follow these guidelines:

Security Consideration	Security Consideration Guidance
User passwords	■ Do not hard-code any passwords in scripts. Using plain text or other unencrypted user credentials is unsafe and can pose a security threat. Whenever possible, use Token-based Authentication (TBA) or OAuth2.0 to specify user credentials. ■ Plaintext passwords can be used with SFTP, certificates, and other areas, however NetSuite always gives the customer the option to use encrypted credential (GUID) instead.
Script Deployment	Deploy scripts to run as administrator only if necessary to minimize security risk and to eliminate performance overhead. See Executing Scripts Using a Specific Role.
Script parameters	For security reasons, do not include confidential information in script parameters. Information saved in script parameters can be indexed by search engines and therefore be viewable by the public. This means, for example, that the information could be found in Google searches. For more information, see Creating Script Parameters Overview.
SuiteScript 2.x RESTlet Script Type	The URLS for accessing RESTlets are protected by TLS encryption. Only requests sent using TLS encryption are granted access. For more information, see the help topic Supported TLS Protocol and Cipher Suites.
SuiteScript 2.x Suitelet Script Type	■ Deploy Suitelets as "Available without Login" only if necessary, such as when there is no user context or to due to login performance overhead. See Setting Available Without Login.) ■ When building custom UI objects outside of the standard UI, such as when building a custom mobile page using a Suitelet), you can use the N/auth Module to help users manage their credentials within the custom UI.
SuiteScript 2.x User Event Script Type	To prevent users from accessing sensitive information, such as password and credit card data, the following internal field IDs cannot be read in `beforeSubmit` entry point scripts for external role users: ■ password ■ password2 ■ ccunumber (on the sales order record) ■ ccsecuritycode (on the sales order record) External role users include shoppers, online form users and other anonymous users, customer center users, etc.
N/auth Module	Take extreme care when using the N/auth module to change an email or user password. Both the auth.changeEmail(options) and auth.changePassword(options) methods allow you to include a plaintext string for the password.
N/http Module and N/https Module	For security purposes, NetSuite blocks some headers from use in HTTP and HTTPS calls. For a list of blocked headers, see the help topic HTTP Header Information and HTTPS Header Information
N/https Module	Plaintext user credentials can be included in HTTPS request parameters or in the body. Using plain text or other unencrypted user credentials is unsafe and can pose a security

Security Consideration	Security Consideration Guidance
	threat. Whenever possible, use Token-based Authentication (TBA) or OAuth2.0 to specify user credentials.
N/query Module	When working with credit cards, you can retrieve only the encrypted version of the credit card number, at most. You may not be able to retrieve and credit card number data.

SuiteScript Debugger

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

NetSuite provides built-in capabilities to enable you to debug your SuiteScript 1.0, SuiteScript 2.0, and SuiteScript 2.1 scripts. In this set of help topics, the debugger used to debug SuiteScript 1.0 and SuiteScript 2.0 scripts is called the *Script Debugger*, while the debugger used to debug SuiteScript 2.1 scripts is called the *2.1 Script Debugger*. To use the Script Debugger or the 2.1 Script Debugger, you must be using a role with SuiteScript permission (Full level).

Note: If you are using SuiteScript 1.0 for your scripts, consider converting these scripts to SuiteScript 2.0 or SuiteScript 2.1 scripts. Use SuiteScript 2.0 to take advantage of new features, APIs, and functionality enhancements. SuiteScript 2.1 includes new language capabilities and functionality to support future editions of the ECMAScript specification, such as support for the spread operator, classes, and destructuring. For more information, see the help topics SuiteScript 2.x Advantages and SuiteScript 2.1.

In This Help Topic

SuiteScript Debugger Overview

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

SuiteScript provides a script debugger for SuiteScript 1.0, SuiteScript 2.0, and SuiteScript 2.1 server scripts, core plug-in implementations, and on-demand debugging.

Important: You cannot use the SuiteScript Debugger to debug SuiteScript 2.1 scripts. You can still test critical parts of your script in the SuiteScript Debugger as a SuiteScript 2.0 script before you run the script as a SuiteScript 2.1 script. For more information about SuiteScript 2.1, see the help topic SuiteScript 2.1.

As a new feature in 2020.2, SuiteScript 2.1 scripts can be debugged with the new 2.1 Debugger. This debugger uses Chrome DevTools directly within NetSuite to allow users to debug their scripts using functionality that is similar to debugging JavaScript in the Google Chrome browser.

Client scripts cannot be debugged using the SuiteScript Debugger, but they can be debugged using the tools available for your browser. To debug client scripts, you should use the Chrome DevTools for Chrome and the Firebug debugger for Firefox. For additional information about these tools, see the documentation provided with each browser.

Note: Currently, only one script can be debugged at a time in a given debug session, regardless of the version of the script.

The following table shows which server script types are supported in each debugger. On-demand debugging is also supported. Client scripts are debugged on the client browser.

Script Type	Script Debugger (SuiteScript 1.0)	Script Debugger (SuiteScript 2.0)	2.1 Script Debugger (SuiteScript 2.1)
Bundle Installation	X	X	—
Map/Reduce	—	—	—
Mass Update	X	X	—
Portlet	X	X	—
RESTlet	X	X	—
Scheduled	X	X	X
SDF Installation	—	This script type runs on the client browser.	—
Suitelet	X	X	X
User Event	X	X	X
Workflow Action	X	X	—
Custom Plug-in	X	X	—

The following table shows which custom plug-ins are supported in each debugger. Support for SuiteScript 2.1 scripts will increase in future releases.

Core Plug-in	Script Debugger (SuiteScript 1.0)	Script Debugger (SuiteScript 2.0)	2.1 Script Debugger (SuiteScript 2.1)
Platform Extension	—	X	Not available in 2021.1
GL Plugin	X	—	
Payment Gateway	X	—	
Consolidated Rate Adjustor	X	—	
Promotions	X	—	
Tax Calculation	X	—	
Shipping Partners	X	X	
Email Capture	X	—	
Advanced Rev Rec	X	X	
Bank Connectivity	X	—	
Test Plugin	X	X	

For more information about using each debugger, see the following help topics:

Debugging Overview

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

This overview section introduces:

Note: To use the Script Debugger and the 2.1 Script Debugger, you must be using a role with SuiteScript permission (Full level).

Debugging Modes

The Script Debugger and the 2.1 Script Debugger both provide two debugging modes:

- On-demand debugging of scripts and code samples: With on-demand debugging, you can debug a new script or a code sample that does not have a defined Script Deployment. Scripts that do not require any form/record-specific interaction are good candidates for on-demand debugging. For more information, see the help topics On-Demand Debugging of SuiteScript 1.0 and SuiteScript 2.0 Scripts and On-Demand Debugging of SuiteScript 2.1 Scripts.
- Debugging deployed scripts: With deployed debugging, you can select an existing script or core plug-in implementation that already has a defined Script Deployment or Plug-in Implementation record. To debug deployed scripts, you must be the script owner, and the status of your script or core plug-in implementation must be set to Testing. For more information, see Debugging Deployed SuiteScript 1.0 and SuiteScript 2.0 Server Scripts and Debugging Deployed SuiteScript 2.1 Server Scripts.

The Debugger Domain

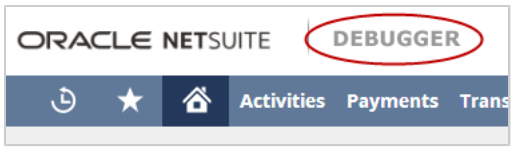
It is important to be aware that when you debug any version of SuiteScript scripts, you are operating in a separate Debugger domain.

Important: Any changes you make to your account when on this Debugger domain **will affect the data in your production account**. For example, if you execute a script in the Debugger that creates a new record, that record will appear in your production account.

Note: You may experience a decrease in performance when working on Debugger domains.

To access the debugger, go to Customization > Scripting > Script Debugger. You can also go directly to the Debugger domain by entering https://debugger.netsuite.com into the browser address bar. The Debugger domain is available from a production account or a release preview account. If you are already logged in to your NetSuite account, you will not need to enter you account information again to access the Debugger domain.

When you are logged in to the Debugger domain, you will see the Debugger logo at the top of the page:



When you are in the Debugger domain, you can debug SuiteScript 1.0, SuiteScript 2.0, and SuiteScript 2.1 scripts if the following requirements are met:

- You must have scripting permission.
- You must be the assigned owner of the script.
- If you are debugging a script that already has a defined script deployment, the script must be in Testing mode before it can be loaded into the debugger. If you want to debug a script that has already been released into production, you must change the script's status from Released to Testing on the Script Deployment page.

The Debugger domain provides access to both the Script Debugger for SuiteScript 1.0 and SuiteScript 2.0 script debugging and to the 2.1 Script Debugger for SuiteScript 2.1 script debugging. You can use either debugger to debug server scripts and custom plug-ins or to debug on demand.

Client scripts can be debugged using the tools available in your browser. Both form-level and record-level client scripts should be tested on the form/record they run against.

If a bundled script has been installed into your account and the script has been marked as hidden, you will not be able to debug this script.

Be aware that the Script Debugger and the 2.1 Script Debugger are not any of the following:

- API test consoles
- Integrated development environments (IDE)
- Script deployment interfaces. To deploy a script to NetSuite, you must still create a script record and define the script's deployment parameters on the Script Deployment page.
- Script runner interfaces (for example, scheduled scripts still need to be placed INQUEUE for task completion)

Terms Related to Debugging

You may find it useful to learn the following terms related to debugging SuiteScript scripts:

Term	Definition
Call Stack	A stack (most recent on top) of all the active functions (and their local variables) called up until the current line of execution.
JavaScript Debugging Pane	The pane, normally on the right side of the Chrome DevTools tab (opened for SuiteScript 2.1 debugging), which displays breakpoints, watches, variables, and the call stack, and includes execution control buttons.
Line Break Point	A user-selected line in source code where program halts execution.
Minify/de-minify	Minified code has had white space and other characters removed from display, so the entire script appears as one long line. This can occur when using the 2.1 Script Debugger. Minified code can be de-minified from within the Debugger. See Using Chrome DevTools.
User Event Break Point	A user event script where program execution should be paused. The user event must be invoked during script execution for the break point to function.

Term	Definition
Watch	A variable or expression that is monitored throughout the program's execution in the current scope.

Debugging SuiteScript 1.0 and SuiteScript 2.0 Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

SuiteScript 1.0 and SuiteScript 2.0 scripts are debugged in the Script Debugger. The Script Debugger enables you to debug SuiteScript 1.0 and SuiteScript 2.0 server scripts and core plug-in implementations, and can be used for on-demand debugging.

Note: To use the Script Debugger and the 2.1 Script Debugger, you must be using a role with SuiteScript permission (Full level).

The following help topics teach you how to use the Script Debugger:

- SuiteScript Debugger Overview
- Script Debugger Interface
- On-Demand Debugging of SuiteScript 1.0 and SuiteScript 2.0 Scripts
- Debugging Deployed SuiteScript 1.0 and SuiteScript 2.0 Server Scripts
- Tips for Debugging SuiteScript 1.0 and SuiteScript 2.0 Scripts

Script Debugger Interface

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

The Script Debugger includes several buttons for controlling the execution of a script and several tabs for viewing data during script execution. The Script Debugger UI includes a script area, buttons and tabs.

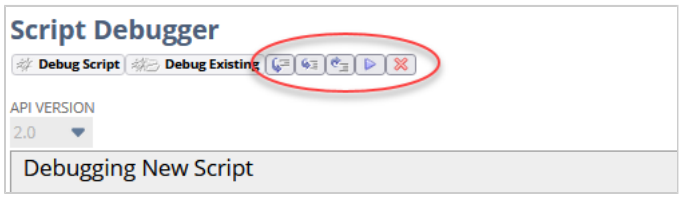
- Script Debugger Buttons
- Script Debugger Tabs

For an example of how to use the Script Debugger buttons and tabs, see Example Use of the Script Debugger Buttons and Tabs.

Script Debugger Buttons

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

The Script Debugger page includes five buttons to allow you to step through and execute your script:



These buttons can be used to control/resume script execution when the debugger stops at a particular line. The following table describes each button. Note that each button also has an associated keyboard shortcut.

Icon	Name	Description	Keyboard Shortcut
	Step Over	Resumes execution from the current line and stops at the next line (even if the current line is a function call).	space
	Step Into	Resumes execution from the current line and stops at the first line in any function call made from the current line.	i
	Step Out	Resumes execution from the current line until the end of the current function, and stops at the first line following the line from where this function was called -or- until the next break point -or- until the program terminates (either by error or by normal completion).	o
	Continue	Resumes program execution from the current line until the next break point -or- until the program terminates.	shift+space
	Cancel	Aborts execution of the program from the current line.	q

Script Debugger Tabs

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

The Script Debugger page includes several tabs to allow you to access information within your script as the script runs:

- Execution Log
- Local Variables
- Watches
- Evaluate Expressions
- Break Points

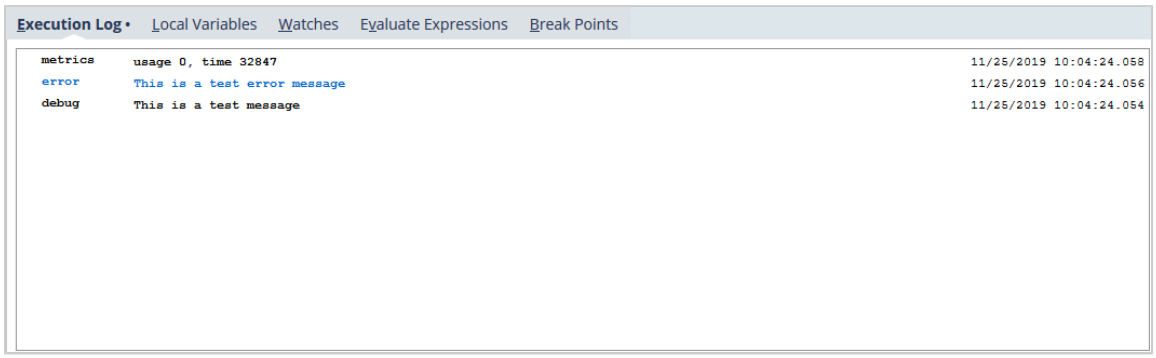
Execution Log

Note: The Execution Log tab is not used for SuiteScript 2.1 scripts. Logging of SuiteScript 2.1 scripts is done on the Chrome DevTools debugger console. For more information about debugging SuiteScript 2.1 scripts, see Introduction to Chrome DevTools for SuiteScript 2.1 Script Debugging.

The Execution Log tab shows all execution logs created by the currently executing program, including errors logged by the system. The execution log details that appear on this tab are the same details that would normally appear in the Execution Log on the Script Deployment page. However, when working in the debugger all script execution details appear on the Execution Log tab on the Script Debugger page; these details will NOT appear on the Execution Log tab of the Script Deployment page.

The type, subject, details, and timestamp are displayed on the Execution Log tab. The timestamp is recorded on the server. It is converted to the current user's time zone for display.

Log details are collapsed by default but can be seen by clicking the expand/collapse icon. Note that the tab is automatically cleared at the start of each debugging session.



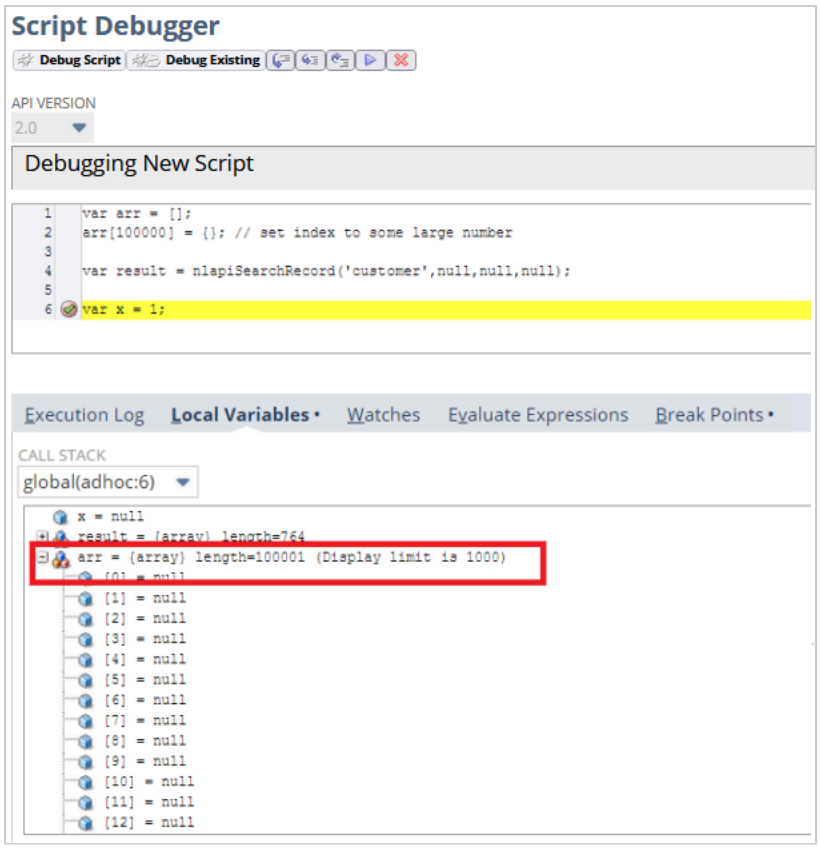
Local Variables

Note: The Local Variables tab is not used for SuiteScript 2.1 scripts. Logging of SuiteScript 2.1 scripts is done on the Chrome DevTools debugger console. See Introduction to Chrome DevTools for SuiteScript 2.1 Script Debugging for more information about debugging SuiteScript 2.1 scripts.

The Local Variables tab shows a list of all local variables (primitives, objects, and NetSuite objects) currently in scope. Note that for NetSuite objects, all properties are private, even though they can be seen on the Local Variables tab. Do not try to reference these properties directly in your script. Use the appropriate getter/setter functions instead.

The Local Variables tab includes a call stack that shows the current execution stack of the program. The function call and current line number for that function are included in the list. Use the Call Stack list to switch to different call stacks to view different local variables. In addition, watch expressions and expression evaluations are automatically performed in the context specified by this field.

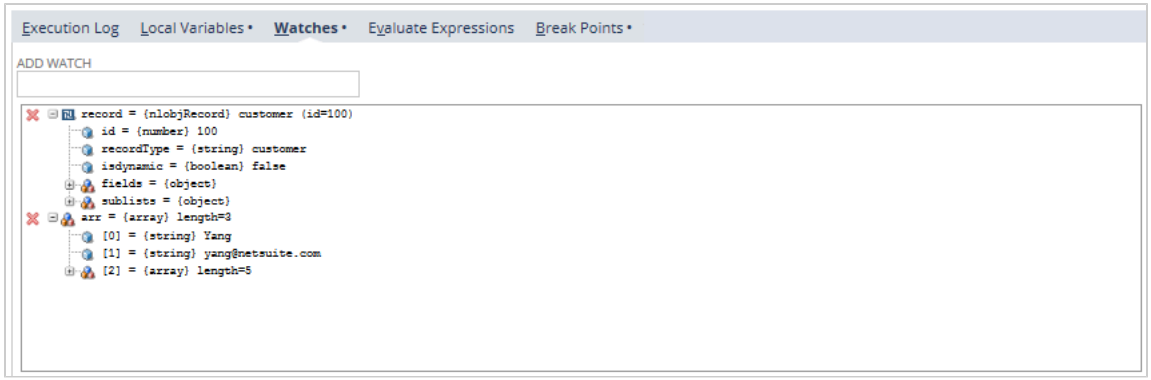
Important: Due to performance considerations, the member display limit for all variables is 1000. In addition, only the first 500 characters of a String are displayed. For large variables, use a watch to see the full member display (see Watches for additional information).



Watches

Note: The Watches tab is not used for SuiteScript 2.1 scripts. Logging of SuiteScript 2.1 scripts is done on the Chrome DevTools debugger console. For more information about debugging SuiteScript 2.1 scripts, see Introduction to Chrome DevTools for SuiteScript 2.1 Script Debugging.

The Watches tab is where you can add or remove variables and expressions to a list that is maintained and kept up-to-date ("watched") throughout the execution of a script.



The variables and expressions are always evaluated in the current call stack. This means that by default they are evaluated at the current line of script execution. However, if you switch to a different function in the call stack, they are re-evaluated at that location.

- To add a variable or an expression, type it into the **Add Watch** field and press the Enter key.
- To remove a watched variable or expression, click on the red **x** icon to the left of the expression.
- To browse sub-properties of a user-defined object, an array, or a NetSuite object expression, click the expand/collapse icon next to the property name.

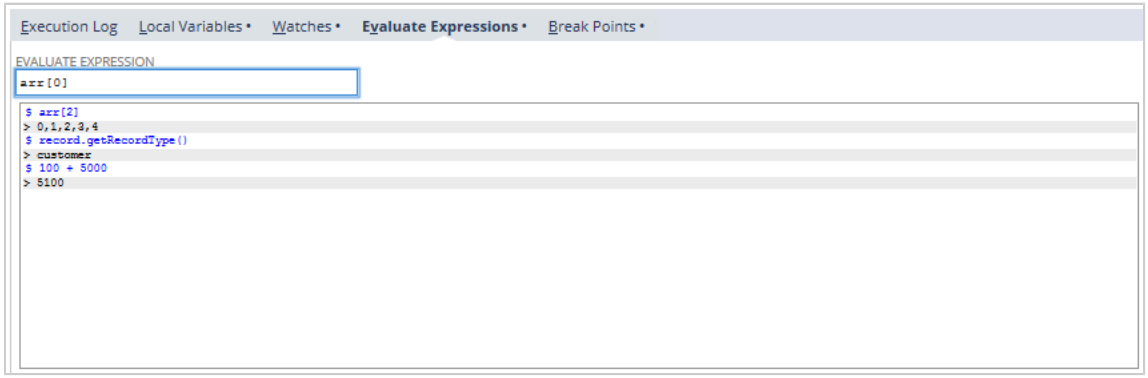
Note that you can use the Watches tab to view object properties using a command-line interface. Any property that is viewable from the property browser can be added as a watch expression by referencing the property using dot (.) notation, even if the property is private in the script. For example, the ID of a record object (referenced by a variable called *record*) by typing record *.id* in the **Add Watch** field.

Evaluate Expressions

Note: The Evaluate Expressions tab is not used for SuiteScript 2.1 scripts. Logging of SuiteScript 2.1 scripts is done on the Chrome DevTools debugger console. For more information about debugging SuiteScript 2.1 scripts, see Introduction to Chrome DevTools for SuiteScript 2.1 Script Debugging.

Use the Evaluate Expressions tab to execute code at break points during the current program. Doing so provides access to the program's state, allowing you to modify the state of the program.

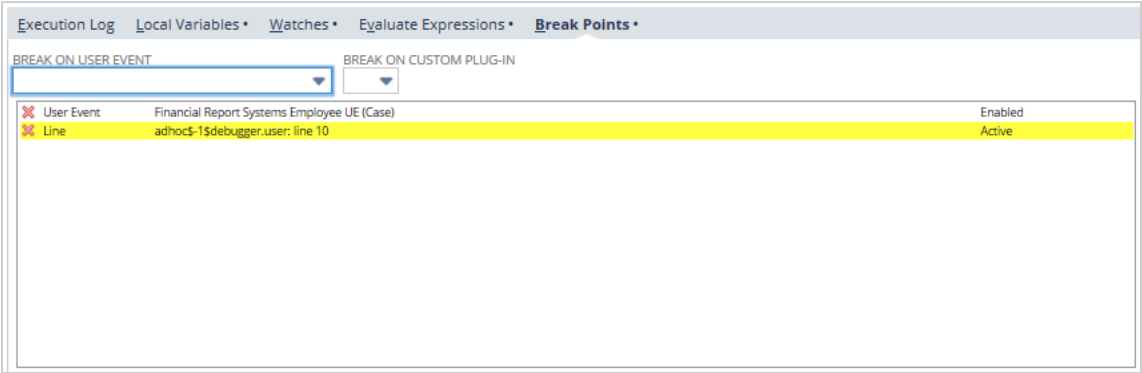
Enter an expression in the Evaluate Expression field and press the Enter key to evaluate the expression at the selected call stack. The results of an evaluated expression (if any) is displayed in the window below. Any change to the program's state is immediately reflected in the Local Variables and Watches tabs.



Break Points

Note: The Break Points tab is not used for SuiteScript 2.1 scripts. Logging of SuiteScript 2.1 scripts is done on the Chrome DevTools debugger console. For more information about debugging SuiteScript 2.1 scripts, see Introduction to Chrome DevTools for SuiteScript 2.1 Script Debugging.

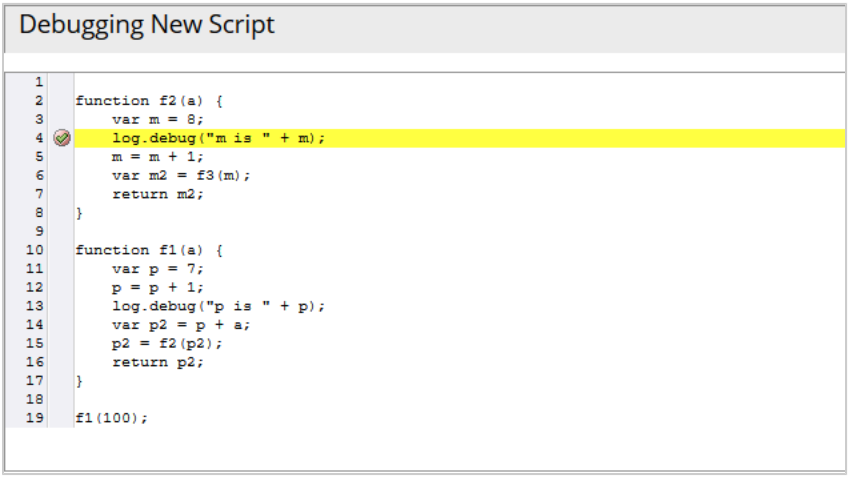
The Break Points tab shows all your instruction-level (line) break points as well as your user event break points. Note that you can add user event break points by selecting user events from the Break on User Event list.



You can set break points in your code using the Script Debugger code window. By setting breakpoints, you can execute your code up to a certain point, and then halt the execution at the break point and examine the current state of the execution.

To add/remove break points in your code:

1. Click between the line number and the line of code to add a breakpoint:



2. To remove a break point, click the break point icon as it appears in the code. You can also remove a break point by clicking the red **x** icon next to the break point, as it appears on the **Break Points** tab.

   Note that when you debug deployed scripts, you can set break points at each user event in your script. User events possibly invokable during script execution where the program halts execution.

Example Use of the Script Debugger Buttons and Tabs

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

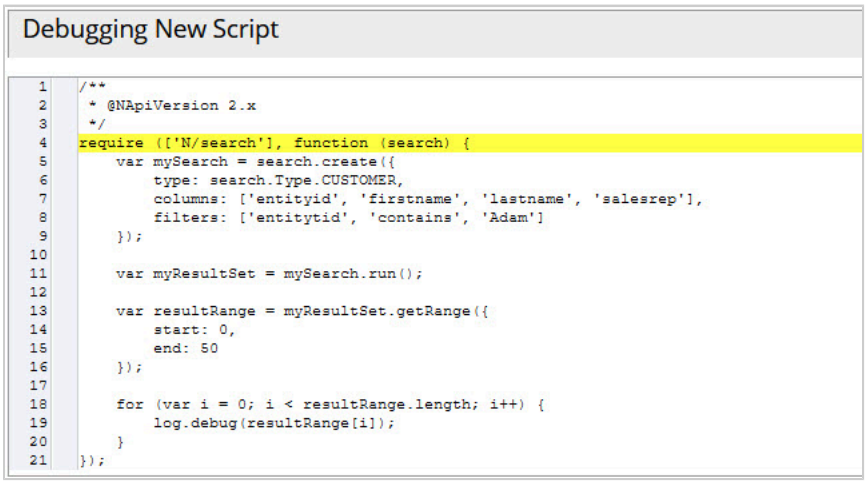
The following example shows how to debug a script using the execute, step over, and step into buttons. This example also shows how to use the Execution Log, Local Variables, and Watches tabs.

1. Go to Customization > Scripting > Script Debugger if you are already logged in to a production or release preview NetSuite account. Or, go directly to the debugger domain by entering https://debugger.netsuite.com into your browser address bar.

2. Enter the following code into the debugger window:

```
/**
 * @NApiVersion 2.x
 */

require (['N/search'], function (search) {
    var mySearch = search.create({
        type: search.Type.CUSTOMER,
        columns: ['entityid', 'firstname', 'lastname', 'salesrep'],
        filters: ['entityid', 'contains', 'Adam']
    });

    var myResultSet = mySearch.run();

    var resultRange = myResultSet.getRange({
        start: 0,
        end: 50
    });

    for (var i = 0; i < resultRange.length; i++) {
        log.debug(resultRange[i]);
    }
});
```

3. In the **API Version** dropdown list, select 2.0.
4. Click **Debug Script**.
5. Wait until the script is shown in the window with the first line highlighted. At this point, the script has not yet started to run.

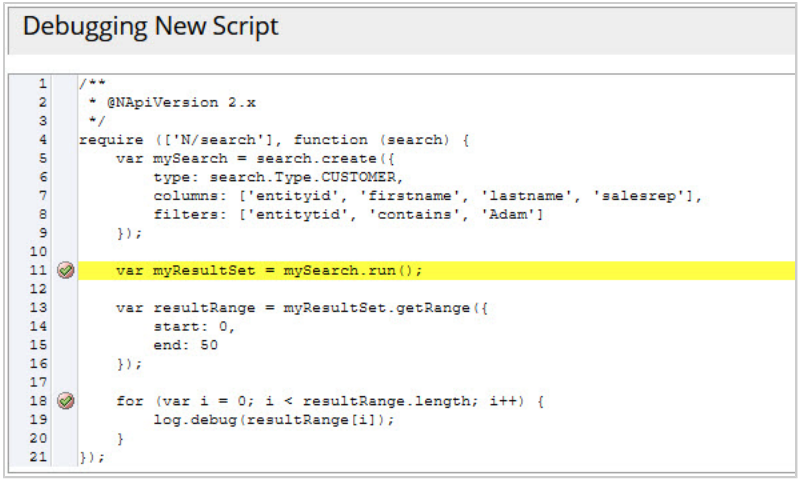


6. Set a breakpoint at the `var myResultSet = mySearch.run()` line by clicking between the line number (11) and the code line. Notice that when you add or remove a breakpoint, "Running Script" is briefly displayed:

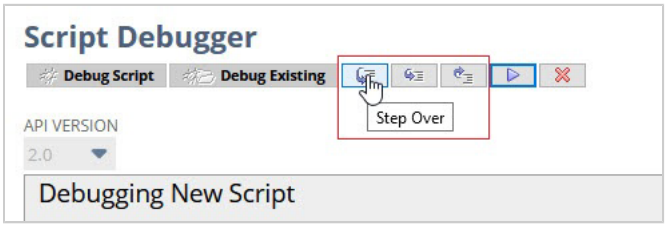
However, the script is not running. But, the breakpoint is being inserted into the proper location.

7. Set another breakpoint at the `for` line (line 18). The two breakpoints are shown like this:

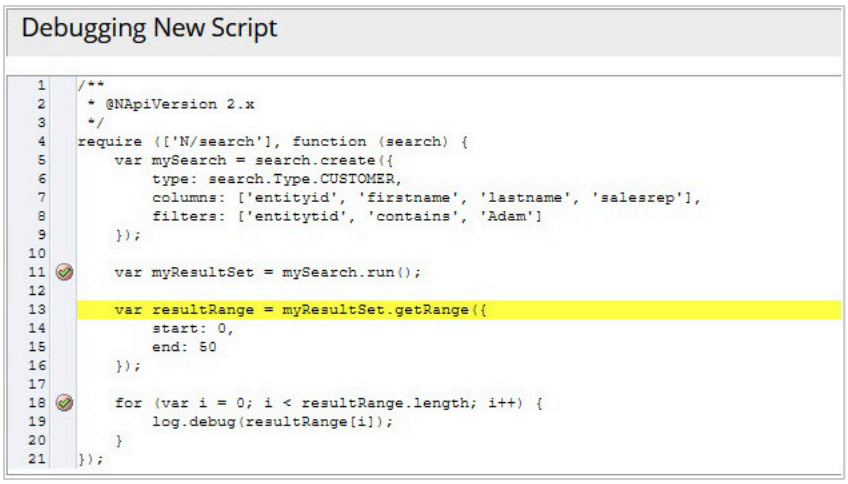
8. Click the Continue button ▷ to execute the script to the first breakpoint.
9. When the breakpoint is reached, the debugger pauses execution and highlights the line:



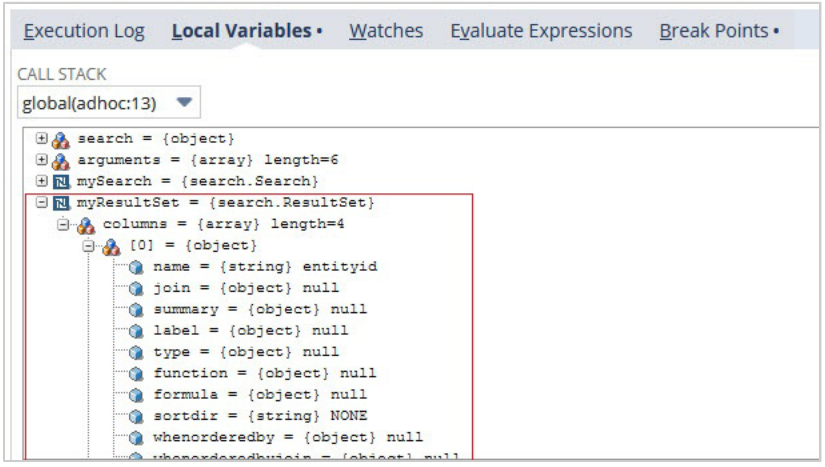
10. Click the Step Over button to execute the `mySearch.run()` line to get your search results:



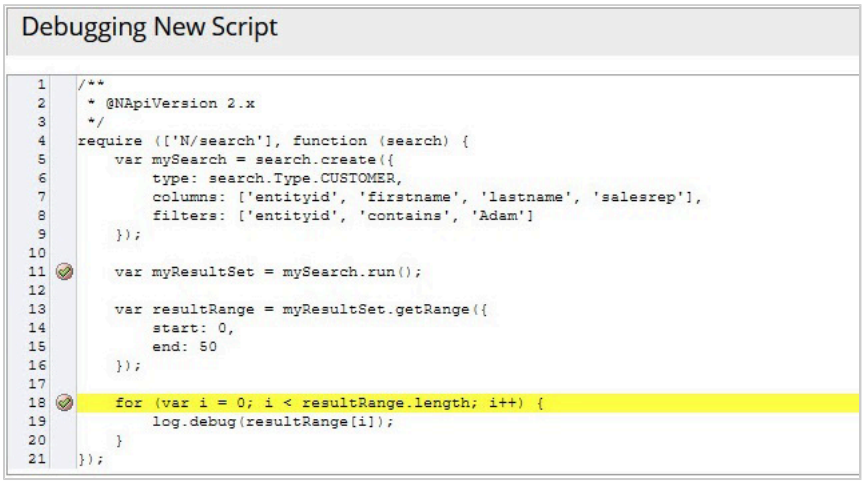
The debugger runs the search and pause at the next line of code:



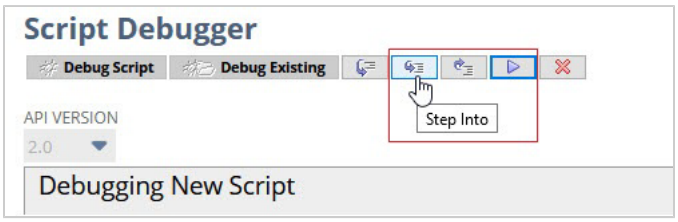
11. While the debugger is paused, click the Local Variables tab to view the value of the `myResultSet` variable:



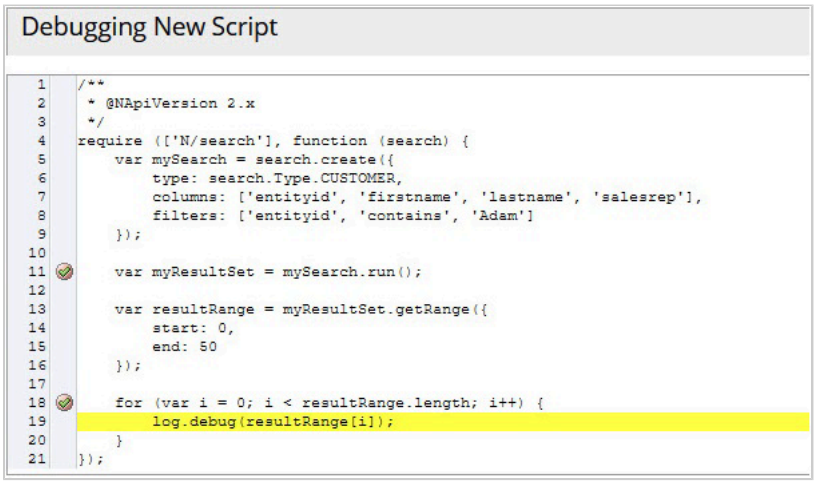
12. Click the Continue button to continue executing the script to the next breakpoint (line 18):



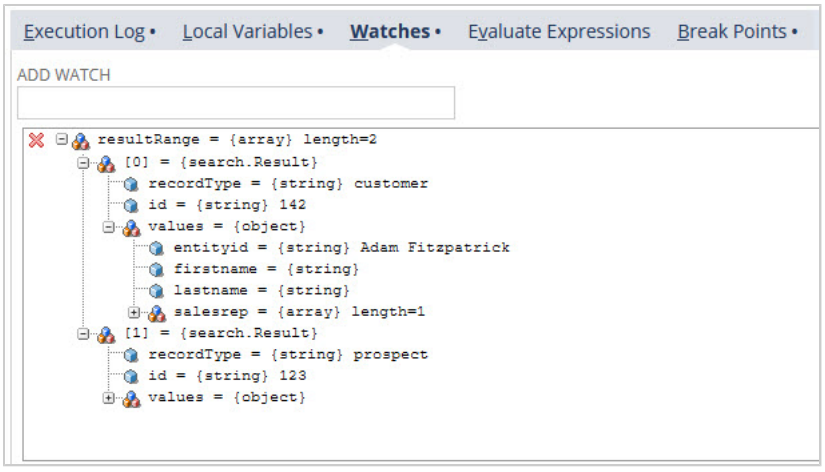
13. When the debugger pauses on line 18, click the Step Into button:



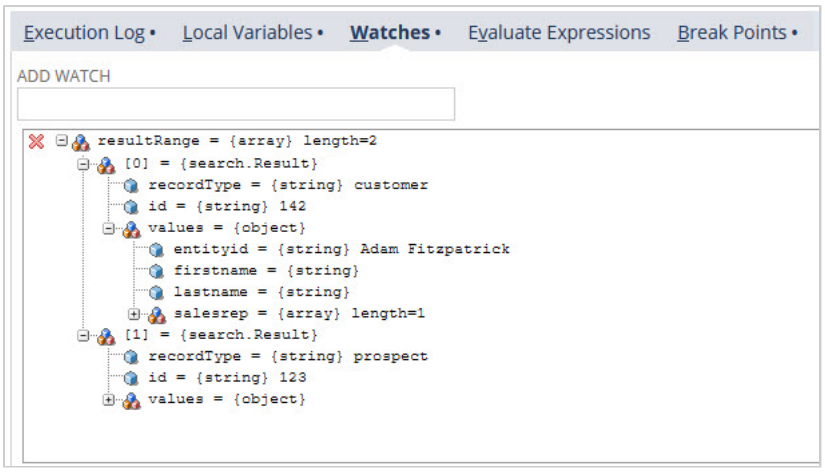
The debugger will execute the `for` line and pause at the first line in the `for` loop block of code, which is the `log.debug` statement:



14. While the debugger is paused, add a watch for the `resultRange` variable to watch the variable value update as the `for` loop. Enter resultRange into the Add Watch box on the Watches tab and press the Enter key to see the value.

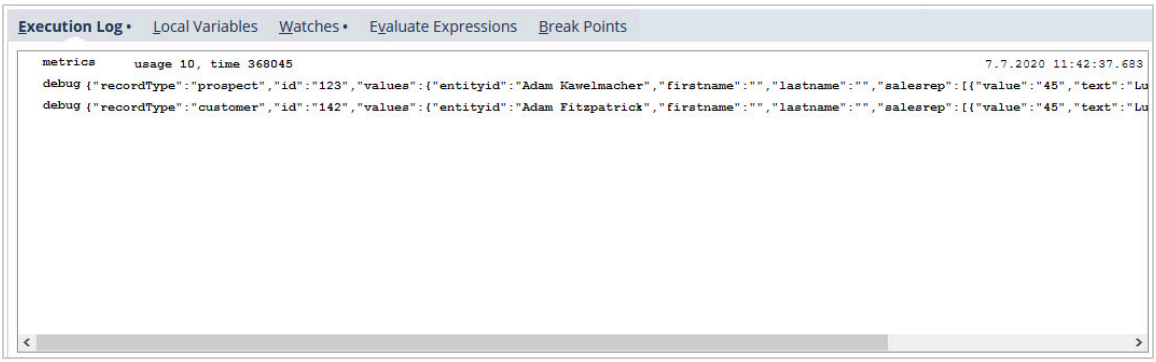


15. Click the Continue button. The debugger executes one iteration of the `for` loop and will again pause.
16. On the Watches tab, expand the `resultRange` variable to view its current value



17. Click the second breakpoint to remove it. You must remove the breakpoint so the debugger does not pause at every iteration of the `for` loop. You can remove any breakpoint at anytime while the debugger is paused.
18. Click the Continue button to finish executing the script.

When the script completes execution, the Execution Log tab shows all you log.debug statements:



On-Demand Debugging of SuiteScript 1.0 and SuiteScript 2.0 Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

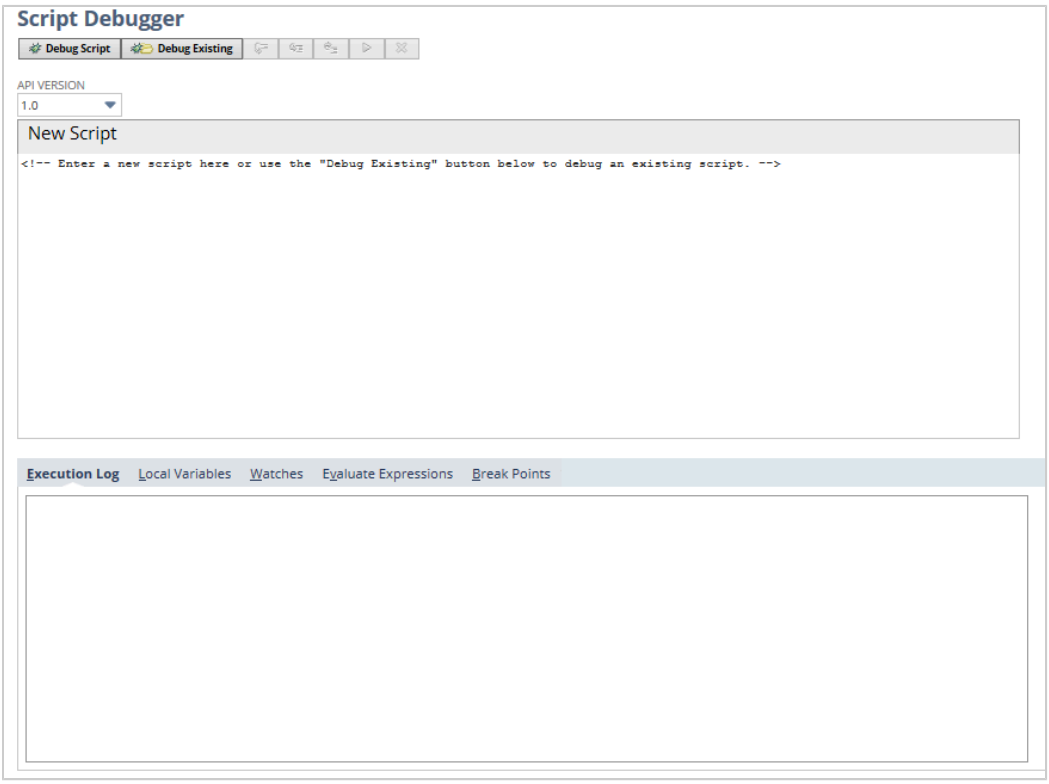
On-demand debugging is used for testing SuiteScript 1.0 or SuiteScript 2.0 scripts or code samples that do not have a defined script deployment. On-demand debugging is **not** used for scripts that have a Script record or defined deployment parameters set on the Script Deployment page. For information about debugging deployed SuiteScript 1.0 and SuiteScript 2.0 scripts, see Debugging Deployed SuiteScript 1.0 and SuiteScript 2.0 Server Scripts.

To use the Script Debugger in on-demand mode:

1. Go to Customization > Scripting > Script Debugger if you are already logged in to a production or release preview NetSuite account. Or, go directly to the Debugger domain by entering https://debugger.netsuite.com into your browser address bar.

Important: Any changes you make to your account when on this Debugger domain **will affect the data in your production account**. For example, if you execute a script in the Debugger that creates a new record, that record will appear in your production account.

The following figure shows the Script Debugger page.



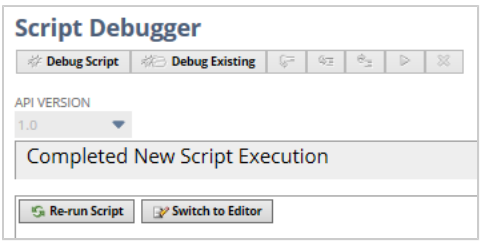
2. Select the **API Version** — 1.0 or 2.0. (If you want to debug a SuiteScript 2.1 script, see On-Demand Debugging of SuiteScript 2.1 Scripts).
3. Enter your code sample or script in the **New Script** text area. If your script is a 2.0 script and includes a `define` statement, you will need to change that to a `require` statement to run the script in the debugger. For more information, see the help topic SuiteScript 2.x Global Objects.

   If you have already written your code in an IDE, copy and paste the code into the **New Script** text area. If you modify your script in the debugger and you intend to save the changes, you must copy the updated script from the debugger and paste it into your IDE.
4. Click **Debug Script**.

   The script is immediately loaded into the debugger and the program's execution stops before running the first line of executable code.

   Important: Make sure you always have an executable call in your script; otherwise, the script has nothing to execute and return.
5. With the script loaded into the debugger, you can step through each line to inspect local variables and object properties. You can also add watches, evaluate expressions, and set break points. See Script Debugger Interface for information about stepping into/out of functions, adding watches, setting and removing break points, and evaluating expressions.

6. When execution of your script is complete, a Completed New Script Execution message appears. You can either re-run the script (by clicking **Re-run Script**), or place the script into edit mode (by clicking **Switch to Editor**) and continue debugging.



Script execution details are logged on the Execution Log tab of the Script Debugger. You can also use the N/log Module within your script to access methods for logging script execution details for all scripts. See Using the Script Execution Log Tab to learn how SuiteScript 1.0 and SuiteScript 2.0 script execution details are logged.

Debugging Deployed SuiteScript 1.0 and SuiteScript 2.0 Server Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

Deployed-mode debugging is for testing scripts or core plug-in implementations that have a defined Script Deployment or Core Plug-in Implementation record.

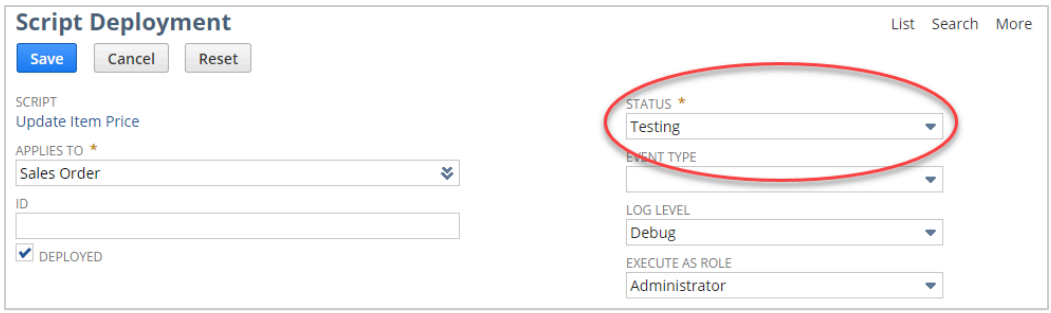
Note: SuiteScript does not support read-only sublists. If you are debugging a script that loads a record with a custom child record as a sublist, make sure the **Allow Child Record Editing** setting is checked for the child record in SuiteBuilder. If this box is not checked, the sublist is read-only and will not load in the parent record. See the help topic Creating Custom Record Types for additional information about creating custom records.

To use the Script Debugger in deployed mode:

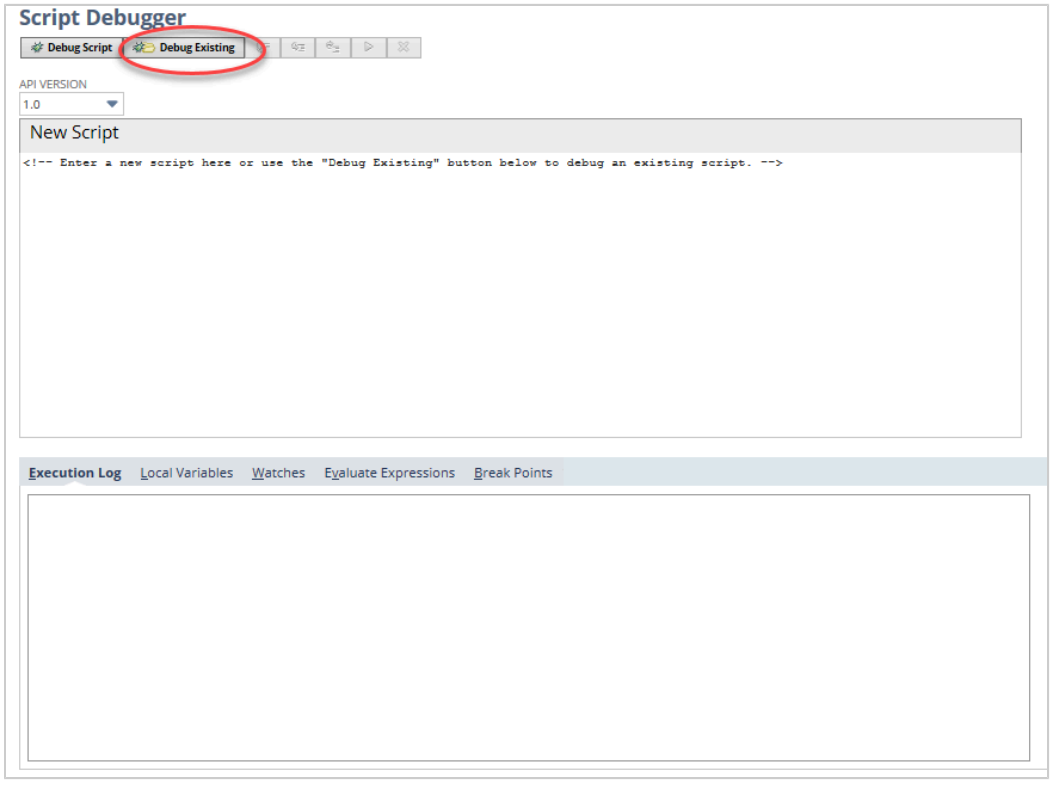
1. Go to Customization > Scripting > Script Debugger if you are already logged in to a production, release preview, or sandbox NetSuite account. Or, go directly to the Debugger domain by entering https://debugger.netsuite.com into your browser address bar.

   Important: Any changes you make to your account when on this Debugger domain **will affect the data in your production account**. For example, if you execute a script in the Debugger that creates a new record, that record will appear in your production account.

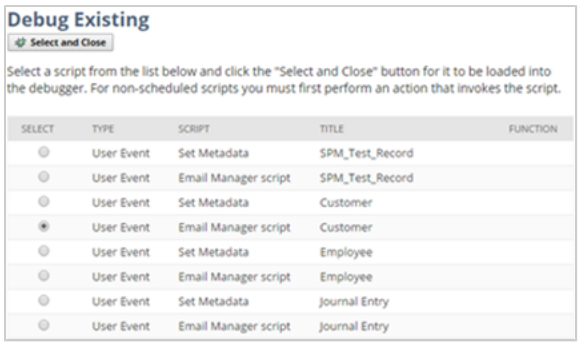
2. When you are on the Debugger domain, go to the applicable Script Deployment or Core Plug-in Implementation record and verify that **Status** is set to Testing. If you want to debug a script or core plug-in implementation that has already been released into production, you must change the status from **Released** to **Testing**.



3. Verify that you are the assigned owner of the script or core plug-in implementation. You can only debug a script or core plug-in implementation if you are the assigned owner.
4. To access the Script Debugger start page, use the appropriate option:
   - If you are on the Script Deployment page in View mode, click **Debug**.
   - If you are on the Script Deployment page in Edit mode, click **Save and Debug**.
   - If you are not on the Script Deployment page or you are debugging a core plug-in implementation, you can access the debugger by going to Customization > Scripting > Script Debugger.
5. On the Script Debugger page, select your **API Version** and then click **Debug Existing**. The list of scripts will be filtered based on the selected API version.



The Debug Existing popup window opens and shows all server scripts and core plug-in implementations available for debugging (see figure below) based on the selected API version. Note that only scripts and core plug-in implementations whose statuses are set to Testing are listed.

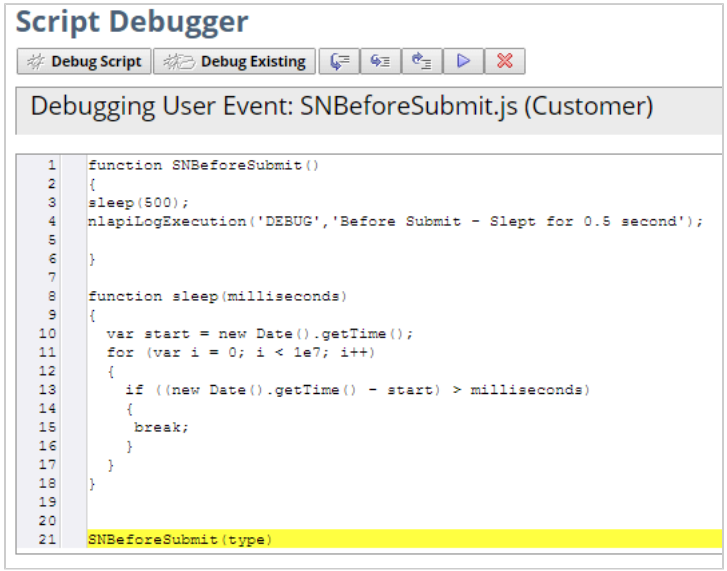


6. Select the script you want to debug and click **Select and Close**.
7. The Script Debugger page is updated with a Waiting for User Action status message. The debugger is waiting for you to perform the action that is associated with the selected script or plug-in implementation.

Note: Adding or updating records using CSV import may not trigger the debugging of scripts because the CSV import may create or update records using a different user role or entity. Remember, that you can only debug scripts if you are the assigned owner.

Note: You do not need to perform any user actions to load scheduled scripts into the Debugger. Simply select your scheduled script from the Script Debugger popup window, and then click Save and Close. The scheduled script automatically loads.

8. When the script is loaded into the debugger, it is shown on the Script Debugger page:



Note: The Debugger does not caption the user action during bulk processing, but the script is executed and a log is created.

Important: You should perfrom user actions in another window or another tab on your browser. This enables you to keep the Script Debugger page open so that you can see the script or core plug-in implementation as it loads. To do this, right-click on the menu item (or other UI element) and select Open Link in New Tab or Open Link in New Window.

With the script loaded into the Debugger, you can now step through each line to inspect local variables and object properties. You can also add watches, evaluate expressions, and set break points. See Script Debugger Interface for information about stepping into/out of functions, adding watches, setting and removing break points, and evaluating expressions.

When execution of a script is complete in the debugger, you can either re-run it (by clicking **Re-run Script**), or place it into edit mode (by clicking **Switch to Editor**) and continue debugging.

Important: If you modify any script or core plug-in implementation in the SuiteScript Debugger and you intend to save the changes, you must copy the updated code from the Script Debugger page and paste it into your IDE (or other editor). You must then re-load the updated file back into the NetSuite File Cabinet.

Script execution details are logged on the Execution Log tab of the Script Debugger. You can also use the N/log Module within your script to access methods for logging script execution details for all scripts. See Using the Script Execution Log Tab to learn how SuiteScript 1.0 and SuiteScript 2.0 script execution details are logged.

Tips for Debugging SuiteScript 1.0 and SuiteScript 2.0 Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

The following tips may help you debug SuiteScript 1.0 and SuiteScript 2.0 scripts using the Script Debugger:

- Review the Execution Log tab on the Script Debugger page to view all script messages and alerts, including those indicating any script errors.
- Execution metrics are shown on the Execution Log tab of the Script Debugger page.
- Use breakpoints to pause script execution at predetermined lines of code. Breakpoints are managed on the Break Points tab on the Script Debugger page. See Break Points for more information.
- Use the Watch tab on the Script Debugger page to see values of variables within your script and to evaluate expressions on-the-fly. See Watches for more information.
- Use the Evaluated Expressions tab on the Script Debugger page to enter and evaluate manually-entered expressions. See Evaluate Expressions for more information.
- To cancel a debugging session without completing script execution, click the **X** button on the Script Debugger page.

Debugging SuiteScript 2.1 Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

As a new feature in 2020.2, SuiteScript 2.1 scripts can be debugged with the new 2.1 Script Debugger. This debugger uses Chrome DevTools directly within NetSuite to allow users to debug their scripts using functionality that is similar to debugging JavaScript in the Google Chrome browser.

Note: To use the Script Debugger and the 2.1 Script Debugger, you must be using a role with SuiteScript permission (Full level).

Note: The 2.1 Script Debugger is fully supported when using NetSuite in the Chrome browser. Other browsers, such as Mozilla Firefox and Apple Safari, have limited or no support. Functionality in those browsers is not guaranteed.

The following help topics teach you how to use the Script Debugger:

- 2.1 Script Debugger Overview
- Introduction to Chrome DevTools for SuiteScript 2.1 Script Debugging
- Using Chrome DevTools
- On-Demand Debugging of SuiteScript 2.1 Scripts
- Debugging Deployed SuiteScript 2.1 Server Scripts
- Debugging Deployed SuiteScript 2.1 Scheduled Scripts and Suitelets
- Debugging Deployed SuiteScript 2.1 User Event Scripts
- Tips for Debugging SuiteScript 2.1 Scripts

2.1 Script Debugger Overview

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

Script debugging is an important part of the holistic developer experience. Debugging with Chrome DevTools is the most common way to debug JavaScript. In our effort to offer a modern coding experience to our developers, we have developed the 2.1 Script Debugger using Chrome DevTools to allow users to debug their scripts using similar functionality available when debugging JavaScript in the Google Chrome browser.

The 2.1 Script Debugger is accessed the same way as the legacy Script Debugger used for SuiteScript 1.0 and SuiteScript 2.0 scripts. To use the 2.1 Script Debugger, select 2.1 as the API Version on the Script Debugger page. When you click Debug Script, a new browser tab opens providing you with the Chrome DevTools interface for debugging your script. Your script will be displayed on the new tab and you will be able to pause/resume execution, set breakpoints, and step through your code line by line.

The 2.1 Script Debugger is available for all SuiteScript developers.

Note: Only SuiteScript 2.1 scripts can be debugged using the 2.1 Script Debugger. You cannot use the 2.1 Script Debugger to debug SuiteScript 1.0 or SuiteScript 2.0 scripts.

In 2021.1, the following SuiteScript 2.1 server script types can be debugged:

- Scheduled scripts
- Suitelets
- User event scripts

Debugging of SuiteScript 2.1 versions of other script types and core plug-in implementations is planned to be supported in a future NetSuite release. You can also debug SuiteScript 2.1 scripts in on-demand mode.

Note: Currently, only one script can be debugged at a time in a given debug session, regardless of the version of the script. You cannot debug a SuiteScript 2.0 script at the same time as a SuiteScript 2.1 script.

The following table lists the debugging capabilities provided by Chrome DevTools along with an indication of whether the capability is supported in the 2.1 Script Debugger in the 2021.1 release. See Using Chrome DevTools for instructions on each capability.

Most common debugging capabilities in Chrome DevTools for JavaScript code	Supported in 2.1 Script Debugger
Pause your code with breakpoints	X
Find unused script code	X (see Chrome DevTools documentation)
Map preprocesses code to source	—
Change thread context	—
Step through code: - step over a line of code - step into/out of a line of code - run all code up to a certain selected line - use breakpoints - restart at the top function of the call stack - pause and resume execution	X
Set and remove breakpoints	X
Edit code	—
View and edit local, closure, and global properties	X
View current call stack	X
Copy stack trace	X (see Chrome DevTools documentation)
Ignore a script or pattern of scripts	X (see Chrome DevTools documentation)
Blackbox a script from the Editor pane, the Call Stack pane, the Settings pane	X (see Chrome DevTools documentation)
Display the console	X

Most common debugging capabilities in Chrome DevTools for JavaScript code	Supported in 2.1 Script Debugger
Use the Memory tab	—
Use the Profiler tab	—
Searching	X
Code coverage	X
Hover over code components shows info and values	X
De-minify the code	X
Set up a workspace	—
Keyboard shortcuts	X See https://developers.google.com/web/tools/chrome-devtools/shortcuts

Introduction to Chrome DevTools for SuiteScript 2.1 Script Debugging

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

The 2.1 Script Debugger provides a subset of the full functionality of the Chrome DevTools available within the Chrome browser.

Note: Only SuiteScript 2.1 scripts can be debugged using the 2.1 Script Debugger. You cannot use the 2.1 Script Debugger to debug SuiteScript 1.0 or SuiteScript 2.0 scripts.

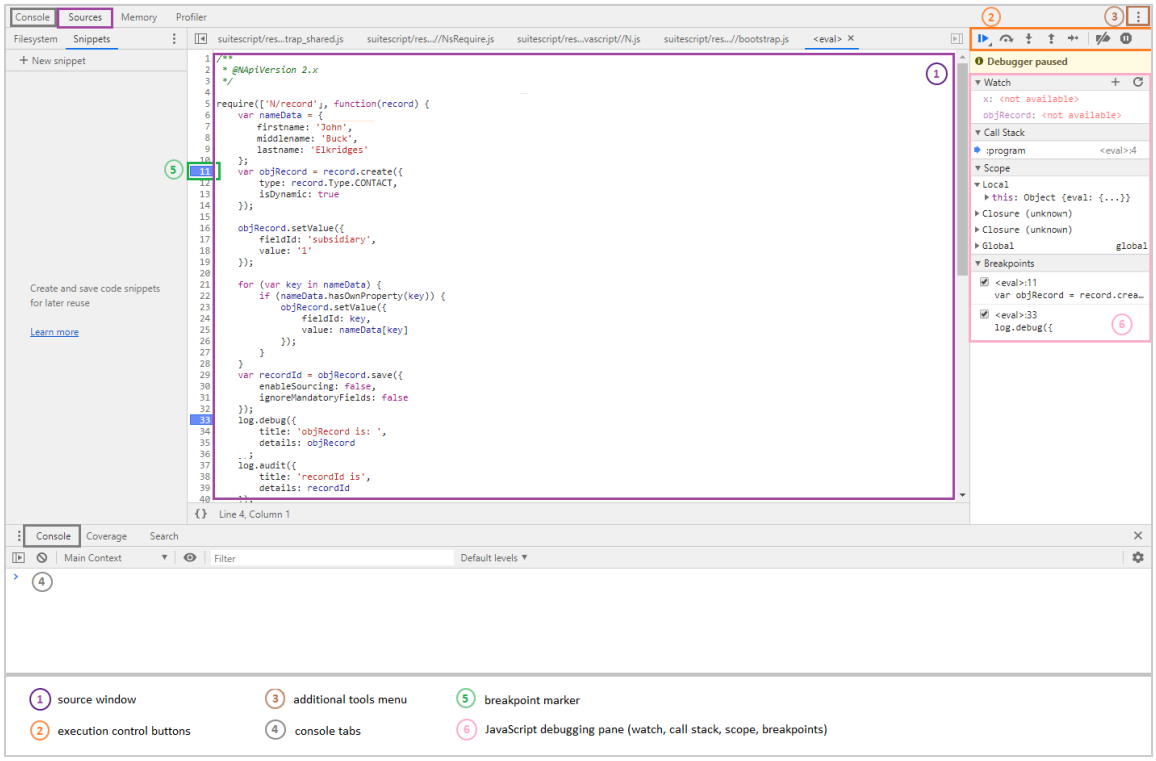
Note: Some of the following information, provided as a summary of the Chrome DevTools as it can be used in debugging JavaScript, is taken from Chrome DevTools. Refer to that site for complete information.

Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. Chrome DevTools can help you edit pages on-the-fly and diagnose problems quickly. The JavaScript portion of the Chrome DevTools site (JavaScript) includes a tutorial and video that teaches you all the things you can do to debug your JavaScript code. Most of these are included in the 2.1 Script Debugger. See the 2.1 Script Debugger Overview section for a list of capabilities provided in the 2.1 Script Debugger.

Tip: Use CTRL+SHIFT+I to access Chrome DevTools from within the Chrome browser. Note that you do not have to do this when debugging SuiteScript because NetSuite automatically provides the Chrome DevTools capability in a separate browser tab.

The following figure shows a sample Chrome DevTools Sources tab opened for debugging SuiteScript 2.1 scripts.

Note: Chrome DevTools for SuiteScript 2.1 scripts supports the Console and Sources tabs only. Currently, there is no support for the Memory or Profiler tabs when debugging SuiteScript 2.1 scripts.

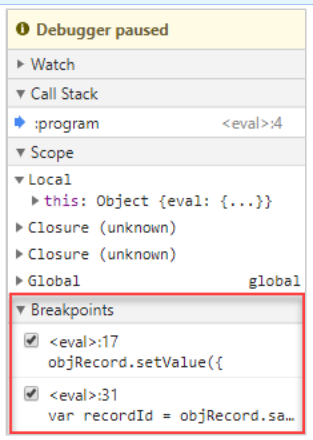
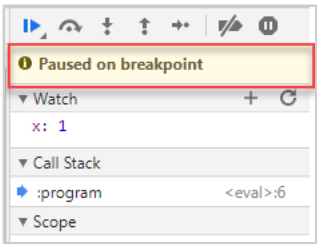
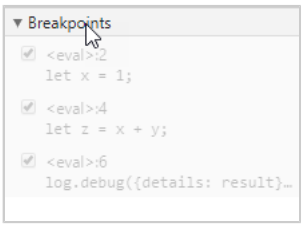


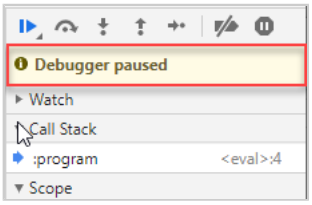
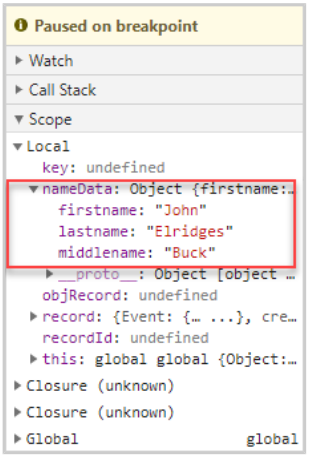
Note: Netsuite's use of Chrome DevTools for SuiteScript 2.1 script debugging adheres to all license parameters as described at Chrome DevTools License.

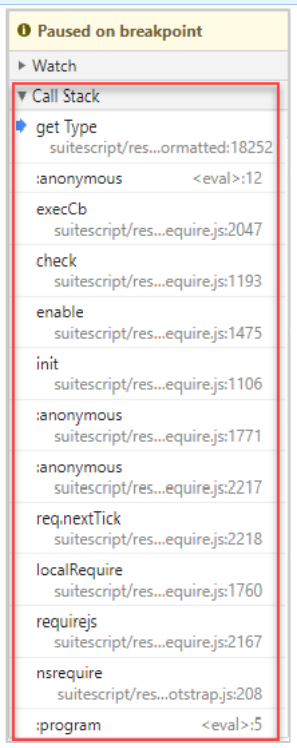
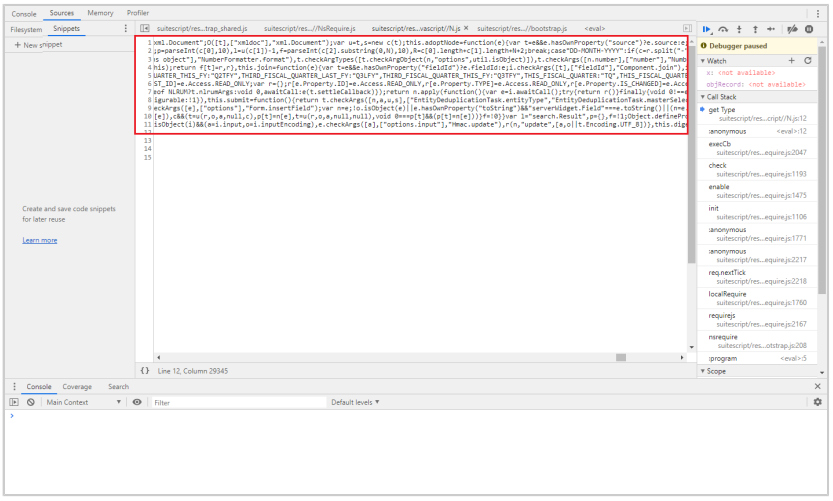
Using Chrome DevTools

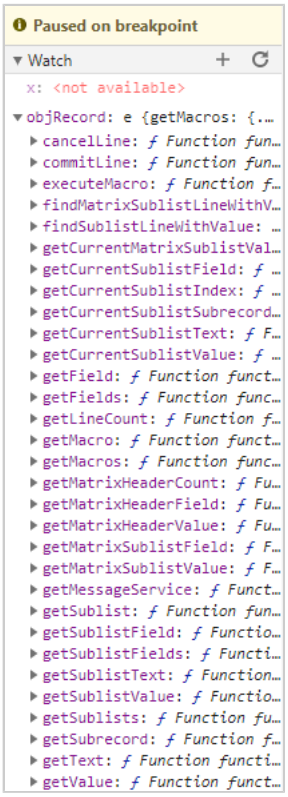
When you start debugging SuiteScript 2.1 scripts using Chrome DevTools, you can set breakpoints, add watches, view the call stack, and start, pause, and resume script execution. Most of these tasks are performed in the 2.1 Script Debugger in the same way they are performed in Chrome DevTools. The following table provides a quick set of instructions for various debug actions.

Debug action	Steps
Set and remove breakpoints	Breakpoints are shown in the Breakpoints section of the JavaScript debugging pane:

Debug action	Steps
	 To set a breakpoint, click to the left of the line of code in the Source window where you want the break to be set. A breakpoint marker is displayed: When execution is paused due to a breakpoint, a Paused on breakpoint message is displayed:  To remove a breakpoint, click the breakpoint marker. **Note:** There may be times when breakpoints "stick" between debugging sessions. If this happens, simply remove the breakpoints that you no longer need.
Disable/re-enable breakpoints	To disable all breakpoints, click the disable breakpoints button:  . A disabled breakpoint marker appears grayed-out:  To re-enable all breakpoints, click the re-enable breakpoints button: . Note that break points must first be set to disable/re-enable them. When all breakpoints are disabled, the breakpoints list is grayed out: 
Step through code	To step over a line of code, click the step over button: . The code pointer moves to the next line.

Debug action	Steps
	To step into a line of code, click the step into button: . The code pointer moves into the function. Note that you cannot step into a SuiteScript module. Clicking the step into button when the code pointer is at a SuiteScript module will move the code pointer to the next line, effectively skipping over the SuiteScript module. To step out of a line of code, click the step out button: . The code pointer moves out of the function to the line after the function call.
Pause and resume execution	To pause execution, click the pause button: . When script debugging is paused, you will see the Debugger paused message:  To resume execution, click the resume execution button:
Restart execution from a line within the call stack	To restart execution from a point in the call stack: 1. Expand the Call Stack section of the JavaScript Debugging pane. 2. Click the line of code in the call stack where you want to resume execution. 3. Click the resume execution button:
View and edit local, closure, and global properties	Values of SuiteScript and JavaScript objects can be viewed by expanding the Local, Closure, or Global sections of the JavaScript debugging pane: 
View current call stack	To view the call stack, click the Call Stack header on the JavaScript debugging pane:

Debug action	Steps
	 You can also select a point in the call stack and resume execution from that point.
Display the console	The console can be displayed in the main debugger window or on the bottom pane of the debugger by clicking the corresponding Console tab. Be aware that if you use the Console tab on the left side of the debugger, the source window closes.
De-minify display of the script code	Sometimes source code will be displayed minified when it is displayed on the Sources tab:  This is often unreadable. You can de-minify the source (that is, add appropriate white space and new lines) by using the {} button located at the bottom of the source window.

Debug action	Steps
Watch variable and expression values	Use the Watch window, on the JavaScript debugging pane, to see values of variables within your script and to evaluate expressions on-the-fly:  To add a variable or enter an expression to evaluate, click the plus (+) button on the Watch window header and enter the variable name or expression. Notice in the above example, the variable being watched is a SuiteScript object so all available members of that object are displayed.

Chrome DevTools supports keyboard shortcuts for nearly all actions available in the debugger. See Chrome DevTools Keyboard Shorcuts. And, your Chrome DevTools environment is retained between debugging session. For instance, if you open the Console window the first time you debug a script, Chrome DevTools will open with the Console window open in each subsequent debugging session.

On-Demand Debugging of SuiteScript 2.1 Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

On-demand debugging is for testing SuiteScript 2.1 scripts or code snippets that do not have a defined script deployment. On-demand debugging is **not** for scripts that have a Script record or defined deployment parameters (set on the Script Deployment page). See Debugging Deployed SuiteScript 2.1 Server Scripts for information about debugging deployed SuiteScript 2.1 scripts.

To debug SuiteScript 2.1 scripts on demand:

1. Go to Customization > Scripting > Script Debugger. You can also go directly to the Debugger domain by entering https://debugger.netsuite.com into the browser address bar.

2. Select 2.1 in the API Version field.
3. Enter you script code into the Debugger window. You may need to include a 'debugger;' statement for the script execution to pause in Chrome DevTools and allow you to gain control prior to script execution. Note that if you use `@NApiVersion` JSDoc tag in your script code, it is essentially ignored. Regardless of the value of the `@NApiVersion` JSDoc tag, your script code will be debugged as a 2.1 script.
4. Click **Debug Script**. A new browser tab opens with the Chrome DevTools debugger and "Waiting for devtools to attach" is displayed on the Script Debugger page above the script. You might also notice that the font of the script in the Debugger window changes slightly, and that the legacy debugger buttons are grayed-out, with the exception of the red X button (that can be used to cancel debugging a 2.1 script). This is an added indication that the Chrome DevTools tab is being activated.

   Note: The first time you use the 2.1 Script Debugger, you may get notification that the popup window was blocked. To use the 2.1 Script Debugger, you must allow this popup window. To do this, click the button on the right side and select to always allow popup windows. You only need to do this one time. All future access to the 2.1 Script Debugger will be immediate.

   Note: When the Chrome DevTools tab opens, it may not show your code. Click the resume script execution button. to access your code. If you code is still not displayed in the Source window, click the :program line under the Call Stack section on the JavaScript debugging pane.
5. You can now debug your code as desired. See the help topic Using Chrome DevTools for information about debug actions you can perform. Use the console to view all log messages. Your code may also be minified. Use the de-minify code button. `{}`) to display your code in a readable manner.

   See Tips for Debugging SuiteScript 2.1 Scripts for additional help when using the Debugger in on demand mode for SuiteScript 2.1 scripts.
6. When execution of your code is complete, "Completed New Script Execution" is displayed on the Script Debugger page above the script. You can close the Chrome DevTools tab at any time after script execution is complete.

Note: When script execution is complete or when an error occurs during script execution, you will see the message: Debugging connection was closed. Reason: Websocket disconnected. This is a normal message. You can close the browser tab and return to the Script Debugger where you can re-run your script or enter a new script for debugging.

Debugging Deployed SuiteScript 2.1 Server Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

Deployed-mode debugging is for testing scripts that have a defined Script Deployment record. Note that plug-in implementations are not supported in 2021.1.

Note: Only SuiteScript 2.1 scripts can be debugged using the 2.1 Script Debugger. You cannot use the 2.1 Script Debugger to debug SuiteScript 1.0 or SuiteScript 2.0 scripts.

In 2021.1, you can debug SuiteScript 2.1 scheduled scripts, Suitelets, and user event scripts using the 2.1 Debugger. See the following help topics for more information:

Debugging Deployed SuiteScript 2.1 Scheduled Scripts and Suitelets

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

You can debug SuiteScript 2.1 scheduled scripts and Suitelets that are in File Cabinet and have a Script Deployment record.

Note: SuiteScript does not support read-only sublists. If you are debugging a script that loads a record with a custom child record as a sublist, make sure the **Allow Child Record Editing** setting is checked for the child record in SuiteBuilder. If this box is not checked, the sublist is read-only and will not load in the parent record. See the help topic Creating Custom Record Types for additional information about creating custom records.

To debug SuiteScript 2.1 scheduled scripts and Suitelets:

1. Create your SuiteScript 2.1 scheduled script or Suitelet, upload it to the File Cabinet, and create a Script Deployment record for it.
2. Go to Customization > Scripting > Script Debugger. You can also go directly to the Debugger domain by entering https://debugger.netsuite.com into the browser address bar.
3. Select 2.1 in the API Version field.
4. Click **Debug Existing Script**. The Debug Existing popup window opens showing all deployed SuiteScript 2.1 scripts.
5. Select your desired deployed script and click **Select and Close**.
6. The Script Debugger page updates displaying "Waiting for User Action" message above the script. This means that it is waiting for you to perform the action associated with how the script is deployed. For example, if you script triggers on the beforeSubmit entry point of a user event script and is deployed on the sales order record, access a sales order and click **Save**.
7. When you perform the necessary action, the "Waiting for devtools to attach" message is displayed above the script on the Script Debugger page and a new browser tab opens for Chrome DevTools.

   Note: The first time you use the 2.1 Script Debugger, you may get notification that the popup window was blocked. To use the 2.1 Script Debugger, you must allow this popup window. To do this, click the button on the right side and select to always allow popup windows. You only need to do this one time. All future access to the 2.1 Script Debugger will be immediate.
8. Your deployed script code is displayed on the new tab where you can debug as desired. A "Debugging <script>.js" message is displayed on the Script Debugger page indicating that the Chrome DevTools tab is ready for use.
9. The Debugger will automatically stop at the first line of your script, which is normally the `define` or `require` statement. You can now debug your code as desired. See the help topic Using Chrome DevTools for information about debug actions you can perform. Use the console to view all log messages. See Tips for Debugging SuiteScript 2.1 Scripts for additional help when debugging scripts.
10. When execution of your code is complete, "Completed Deployed Script Execution" is displayed on the Script Debugger page above the script. You can close the Chrome DevTools tab at any time after script execution is complete.

Note: When script execution is complete or when an error occurs during script execution, you will see the message: Debugging connection was closed. Reason: Websocket disconnected. This is a normal message. You can close the browser tab and return to the Script Debugger where you can re-run your script or enter a new script for debugging.

Debugging Deployed SuiteScript 2.1 User Event Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

SuiteScript 2.1 user event scripts can be debugged using the 2.1 Debugger similar to other supported server script types (scheduled scripts and Suitelets) as described in Debugging Deployed SuiteScript 2.1 Scheduled Scripts and Suitelets. However, because a user event script can have multiple entry points, there are some special steps to take when debugging SuiteScript 2.1 user event scripts.

You can debug every entry point included in your SuiteScript 2.1 user event script within the same browser session, however, you must reattach to Chrome DevTools for each entry point included in the script.

Note: SuiteScript does not support read-only sublists. If you are debugging a script that loads a record with a custom child record as a sublist, make sure the **Allow Child Record Editing** setting is checked for the child record in SuiteBuilder. If this box is not checked, the sublist is read-only and will not load in the parent record. See the help topic Creating Custom Record Types for additional information about creating custom records.

Following is an example of debugging a user event script deployed on the Customer record. You first need to create the user event and start the debugger before you can debug the script.

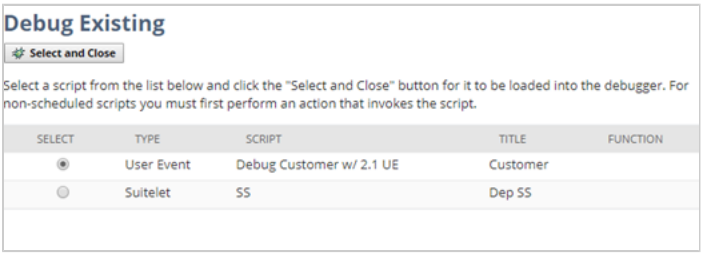
To create the user event script and start the debugger:

1. Create your user event script file and upload it to the File Cabinet. An example script is shown here:

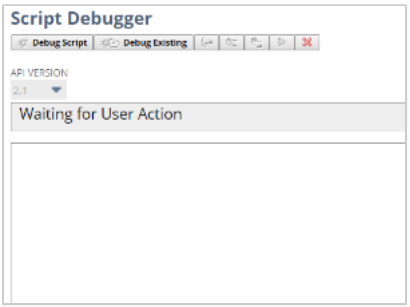
```
/**
 * @NApiVersion 2.1
 * @NScriptType UserEventScript
 */
define (['N/record'], function (record) {
    function beforeLoad(context) {
        if (context.type !== context.UserEventType.CREATE)
            return;

        var customerRecord = context.newRecord;

        customerRecord.setValue('phone', '555-555-5555');

        if (!customerRecord.getValue('salesrep')) {
            customerRecord.setValue('salesrep', 59); // replace 59 with a value specific to your account
        }
    }

    function beforeSubmit(context) {
        if (context.type !== context.UserEventType.CREATE)
            return;

        var customerRecord = context.newRecord;

        customerRecord.setValue('comments', 'Please follow up with this customer!');
    }

    function afterSubmit(context) {
        if (context.type !== context.UserEventType.CREATE)
            return;
```

```
31
32          var customerRecord = context.newRecord,
33
34          if (customerRecord.getValue('salesrep')) {
35              var call = record.create({
36                  type: record.Type.PHONE_CALL,
37                  isDynamic: true
38              });
39
40              call.setValue('title', 'Make follow-up call to new customer');
41              call.setValue('assigned', customerRecord.getValue('salesrep'));
42              call.setValue('phone', customerRecord.getValue('phone'));
43
44              try {
45                  var callId = call.save();
46                  log.debug('Call record created successfully', 'Id: ' + callId);
47              } catch (e) {
48                  log.error(e.name);
49              }
50          }
51      }
52
53      return {
54          beforeLoad: beforeLoad,
55          beforeSubmit: beforeSubmit,
56          afterSubmit: afterSubmit
57      }
58  }
```

2. Create a script record for your script. For this example, set the Name field to Debug Customer w/ 2.1 UE and set the Applies To: field to Customer.
3. To have the script appear on the debugger tab, it must be executed by creating and saving a new Customer record (List > Relationships > Customers > New).
4. Go to Customization > Scripting > Script Debugger.
5. Select Debug Existing and select the Debug Customer w/ 2.1 UE user event script.



6. Click Select and Close. The Script Debugger will wait for a user action to proceed:

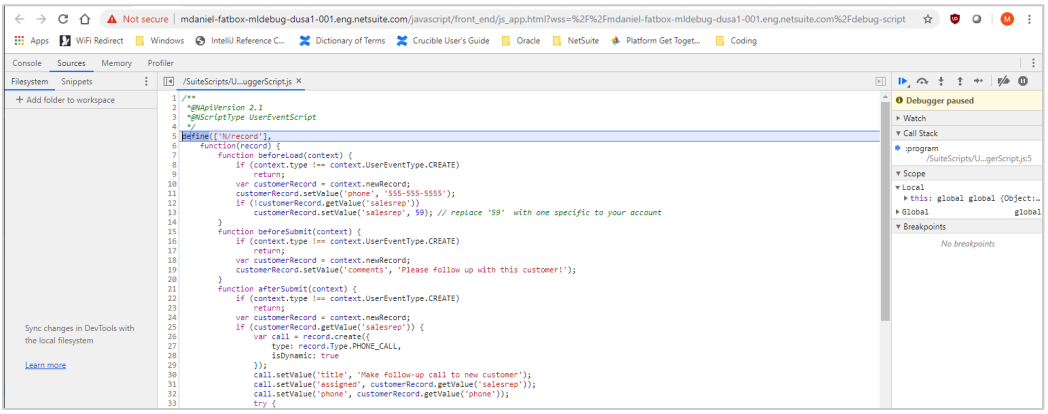


7. Follow specific instructions for debugging each entry point.

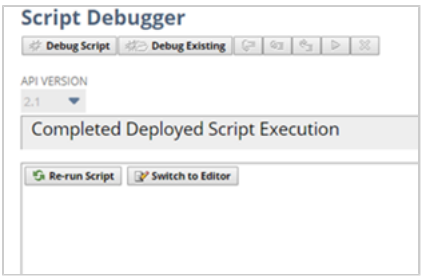
To debug the beforeLoad entry point:

1. Follow the steps above to create the user event script and start the debugger.

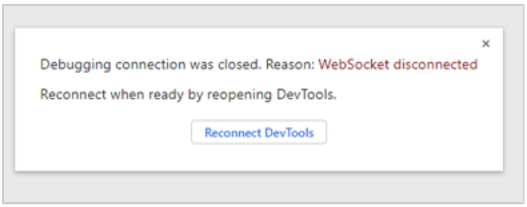
2. To trigger the beforeLoad entry point, either create a new Customer or edit an existing Customer.
3. When the customer record is loaded, a new Chrome DevTools tab opens in your browser showing your user event script ready to be debugged using the 2.1 Debugger:



4. The execution of your script stops at the top of the script. You can now add breakpoints, watches, etc. and begin debugging your script. You can also add a 'debugger;' statement at the top of the beforeLoad entry point code to specifically stop the execution of the script at that entry point.
5. When you have finished stepping over/playing through the last line of the beforeLoad entry point code, the Script Debugger tab will show:



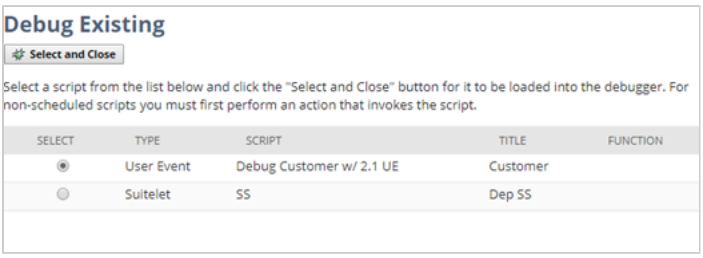
And you will see a message in the Chrome DevTools tab that indicates the debugging connection was closed:



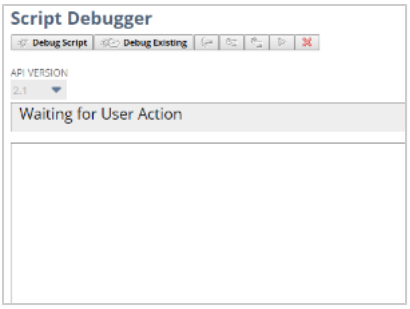
You do not need to click the Reconnect DevTools button on this message. If you want to reexecute the beforeLoad entry point in your script, save the Customer record to restart the debugging session

To debug the beforeSubmit and afterSubmit entry points:

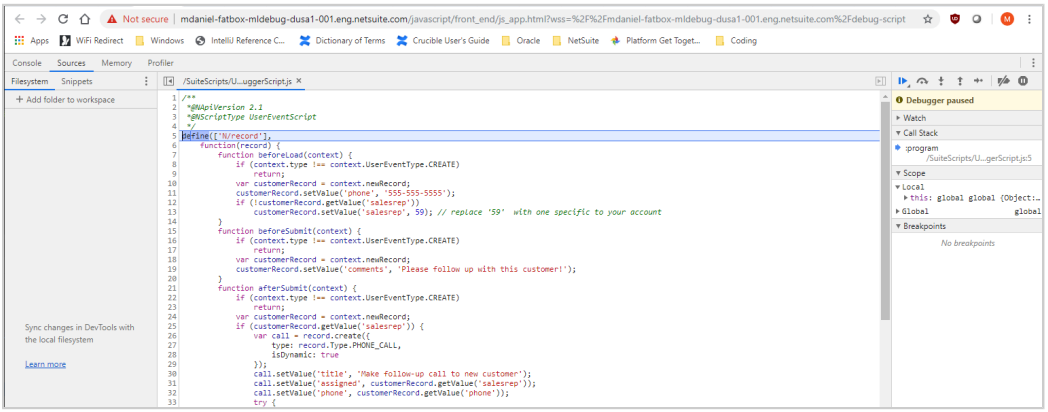
1. Go to Customization > Scripting > Script Debugger, or re-execute the script if you are already in the Script Debugger from debugging the beforeLoad entry point.
2. Create a new Customer or edit an existing Customer. **Do not save the record yet**. You first need to get ready to debug the script, as shown in the next steps, before clicking Save.
3. Select Debug Existing and select the Debug Customer w/ 2.1 UE user event script:



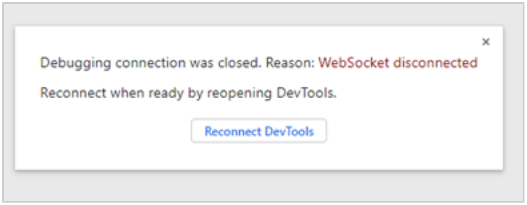
4. Click Select and Close. The Script Debugger will wait for a user action to proceed:



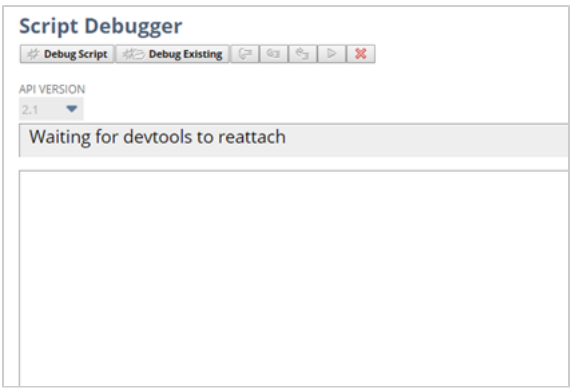
5. To trigger the beforeSubmit entry point, save the Customer record. A new Chrome DevTools tab opens in your browser showing your user event script ready to be debugged using the 2.1 Debugger:



6. The execution of your script stops at the top of the script. You can now add breakpoints, watches, etc. within the beforeSubmit entry point and begin debugging your script. You can also add a 'debugger;' statement at the top of the beforeSubmit and afterSubmit entry point code to specifically stop the execution of the script at that entry point.
7. After the last line of the beforeSubmit function executes, the following message will appear in the Chrome DevTools tab:

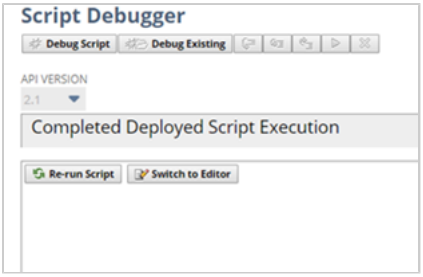


And the Script Debugger shows:

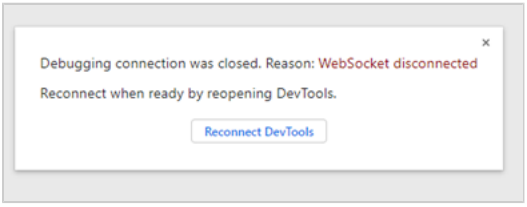


Note: This reattach phase only appears if you have both a beforeSubmit and an afterSubmit entry point in your user event script. If you have only the beforeSubmit entry point or the afterSubmit entry point, your debugging session will end at the completion of the entry point code.

8. To reattach the debugger and continue to debug your afterSubmit entry point code, click the Reconnect DevTools button on the message. The Chrome DevTools tab will reload your script and execution will stop at the beginning of the afterSubmit entry point code. You can now add breakpoints, watches, etc. and begin debugging your script.
9. When you have finished stepping over/playing through the last line of the afterSubmit entry point code, the Script Debugger tab will show:



And you will see a message in the Chrome DevTools tab that indicates the debugging connection was closed:



You do not need to click the Reconnect DevTools button on this message. If you want to re-execute the any entry point in your script, you will need to reload a customer record and follow the steps to debug the beforeLoad entry point or to debug the beforeSubmit and afterSubmit entry points.

Tips for Debugging SuiteScript 2.1 Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

Note: Only SuiteScript 2.1 scripts can be debugged using the 2.1 Script Debugger. You cannot use the 2.1 Script Debugger to debug SuiteScript 1.0 or SuiteScript 2.0 scripts.

The following tips may help you debug SuiteScript 2.1 scripts using the Chrome DevTools debugger:

- If your script code appears minified (lacking white space and new line characters), you can use the de-minify button. `{}` to change the display of the code so that it is readable.
- Details normally displayed on the Execution Log of the Script Debugger page are shown on the Chrome DevTools console when debugging SuiteScript 2.1 scripts.
- Execution metrics for SuiteScript 2.1 scripts are shown in the Chrome DevTools console when the script completes execution. These are general the same metrics displayed for SuiteScript 1.0 and SuiteScript 2.0 scripts in the Script Debugger page and include script messages and alerts, including script errors.
- Use breakpoints to pause script execution at predetermined lines of code. Active breakpoints are listed in the Breakpoints List in the JavaScript debugging pane (on the Chrome DevTools tab).
- Use the Watch window on the JavaScript debugging pane to see values of variables within your script and to evaluate expressions on-the-fly.
- Use the Call Stack window on the JavaScript debugging pane to view different areas of your script. You can also select a point in the call stack and resume execution from that point.
- To cancel a debugging session without completing script execution, click the **X** button on the Script Debugger page. The Chrome DevTools tab can be left opened or it can be closed. If you leave it open, it will be used for the next debugging session. Note that simply closing the tab does not fully cancel the current debugging session. You must use the red X button.
- Each 2.1 debugging session has a timeout. If you do not perform any actions in the Chrome DevTools debugger within the timeout period, your debug session will end. See Idle Timeouts.

Debugging Client Scripts

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

Client scripts are executed on the client browser based on how they are deployed. To debug a client script, use the debugging tools available on your browser. SuiteScript supports several browsers:

- Google Chrome (preferred)
- Mozilla Firefox (preferred)
- Microsoft Edge
- Apple Safari

Each browser includes its own debugging tools. The following table includes links to documentation for each browser.

Browser	Documentation Link
Google Chrome	https://developers.google.com/web/tools/chrome-devtools/javascript
Mozilla Firefox	https://developer.mozilla.org/en-US/docs/Tools/Debugger
Microsoft Edge	https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide/debugger
Apple Safari	https://developer.apple.com/library/archive/documentation/AppleApplications/Conceptual/Safari_Developer_Guide/Debugger/Debugger.html and https://support.apple.com/guide/safari-developer/welcome/mac

Client scripts (both form- and record-level) should be tested on the form/record they run against. See the help topic SuiteScript 2.x Client Script Entry Points and API for more information about triggering a client script. Also see the help topic Record-Level and Form-Level Script Deployments for more information about deploying a client script.

All browsers support common actions within their debugging environments. These actions including stepping through or over code, setting breakpoints, and pausing/resume execution.

Important: To debug a client script, the script MUST include a 'debugger;' statement. Place the 'debugger;' statement near the top of the script so that the debugger is invoked immediately when the script is triggered. Execution will stop when the statement is reached, allowing you to examine script properties and variables using the debugging tools in your browser.

To debug a SuiteScript client script:

1. Create, upload and deploy your client script.
2. Get ready to perform the action to trigger your client script. For example, if your client script trigger (entry point) is saveRecord on the Purchase Order record, open a new purchase order in NetSuite, enter your data, but do not click **Save**.
3. Access the debugger in your browser (before triggering the script):
   - Enter CTRL+SHIFT+I in Chrome
   - Enter CTRL+SHIFT+I in Firefox
   - Enter CTRL+OPTION+I in Safari

   Note: If you access the browser debugger before you are ready to trigger your script, the debugger may pause at every action you perform in the browser causing unnecessary and premature interaction with the debugger.
4. When the debugger is open, click **Save** on the purchase order (in this example). The debugger will pause the execution of your script at the location of the ' debugger;' statement. You can now use the debugger tools in your browser to step through the code, look at values, set breakpoints, etc. Refer to the documentation for your browser for more information.

   When debugging client scripts, some scripts might be minified. Minified scripts have all unnecessary characters removed, including white space and new line characters. You can use your browser debug tools to de-minify the script. This is typically done using the { } button. See the documentation for your browser for more information.
5. When you are done using the debugger, it can be closed. Your NetSuite page will remain displayed and you can continue using NetSuite.

For additional information about working with client scripts, see the help topics SuiteScript 2.x Client Script Entry Points and API and Record-Level and Form-Level Script Deployments.

Debugging a RESTlet

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

You can use the NetSuite Debugger to debug RESTlet code in the same manner that you debug other types of SuiteScript code, as described in Debugging Deployed SuiteScript 2.1 Server Scripts.

If you have installed and set up SuiteCloud IDE, a debug client is available for your use. The RESTlet/Suitelet Debug Client enables you to debug deployed RESTlet and Suitelet SuiteScripts with the SuiteCloud IDE Debugger. The client is only accessible after a debug session is started. For information about SuiteCloud IDE, see SuiteCloud IDE Plug-in for Eclipse Guide.

Note: In addition to debugging RESTlet script code, you should test the HTTP request to be sent to the RESTlet. Free tools are available for this purpose, such as Send HTTP Tool (https://www.softpedia.com/get/Internet/Servers/Server-Tools/WIN-HTTP-Sender.shtml).

To debug a deployed RESTlet:

1. Before you deploy a RESTlet to be debugged, ensure that the script does not include the HTTP authorization header, as this header can prevent the debugger from working.
2. Ensure that on the script deployment record, the **Status** value is set to **Testing**.
3. Go to Customization > Scripting > Script Debugger, or log in to the debugger domain **https://debugger.netsuite.com**.
4. Click the **Debug Existing** button in the main Script Debugger page.

   Note: This button only appears when you have deployed scripts with the status is set to **Testing**.
5. Select the RESTlet script that you want to debug in the Script Debugger popup.

   After you click the **Select** option button, the RESTlet's cookies display in a banner.
6. Copy the cookies and paste them into a text file so that you have them available.
7. Click the **Select and Close** button in the Script Debugger popup.

   The main Script Debugger page displays a message that it is waiting for user action.
8. Set the cookies in your client application to the values you copied in step 6, and send the RESTlet request.

   The main Script Debugger page displays the script execution as pending at the NetSuite function `restletwrapper(request)`.
9. With the script loaded into the Debugger, you can now step through each line to inspect local variables and object properties. You can also add watches, evaluate expressions, and set break points. See Script Debugger Interface for information about stepping into/out of functions, adding watches, setting and removing break points, and evaluating expressions.

Related Topics

- SuiteScript Debugger
- Debugging Overview
- Debugging SuiteScript 1.0 and SuiteScript 2.0 Scripts
- Debugging SuiteScript 2.1 Scripts

Script Debugger Metering and Permissions

Applies to: SuiteScript 1.0 | SuiteScript 2.x | SuiteCloud Developer

The Script Debugger adheres to the following metering and permission restrictions for all SuiteScript 1.0 and SuiteScript 2.0 script types:

The Script Debugger adheres to the following metering and permission restrictions for all SuiteScript 1.0, SuiteScript 2.0, and SuiteScript 2.1 script types:

- A user is only allowed to debug one script at a time. Attempting to debug multiple scripts simultaneously (for example, by opening two different browser windows) will result in the same script/debugging session appearing in both windows.
- Users can debug only their own scripts in their current login session.
- There is a 1000 unit usage limit on all scripts being debugged. This is important to note, particularly for script types such as scheduled scripts, which are permitted 10,000 units when running in NetSuite. If, for example, you load a 2,000 unit scheduled script into the Debugger and attempt to step through or execute your code, the Debugger will throw a usage limit error when it reaches 1000 units.
- Email error notification is disabled for scripts being debugged.
- Execution log details are displayed on the Execution Log tab in the Debugger rather than in the execution log on the Script Deployment page for all SuiteScript 1.0 and SuiteScript 2.0 scripts.
- Execution log details are displayed on the Execution Log tab in the Debugger rather than in the execution log on the Script Deployment page for all SuiteScript 1.0 and SuiteScript 2.0 scripts. For SuiteScript 2.1 scripts, execution log details are displayed on the Console tab of the Chrome DevTools debugging window.

Also refer to the SuiteScript Governance and Limits help topic for governance limits on script types and API modules.

Idle Timeouts

Both the SuiteScript 1.0 and 2.0 Debugger and the 2.1 Script Debugger have idle timeouts, where your debug session will end if you stop interacting with the debugger, or if you have exceeded the maximum allowed amount for a debug session.

SuiteScript 1.0 and SuiteScript 2.0

There is a two-minute time limit on SuiteScript 1.0 and 2.0 scripts sitting idle in the debugger. If you do not perform some user action in the debugger within the two minutes, the following error is thrown:

- You have exceeded the maximum allowable idle time for debugging scripts. To debug another script, reload the script debugger page and start a new debugging session.

If you receive this message and you are debugging a deployed SuiteScript 1.0 or SuiteScript 2.0 script, click **Go Back**. You must then reload your script by clicking **Debug Existing**.

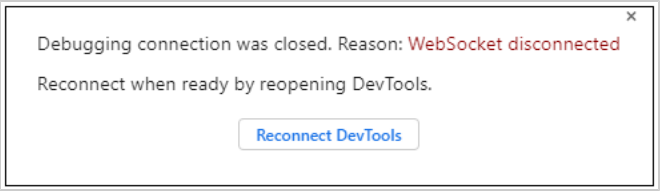
If you are debugging a SuiteScript 1.0 or SuiteScript 2.0 script in on-demand mode, click **Go Back**, click in the Debugger text area, and re-type (or copy and paste) your code snippet. See Debugging SuiteScript 1.0 and SuiteScript 2.0 Scripts for information about deployed and on-demand debugging modes for SuiteScript 1.0 and SuiteScript 2.0 scripts.

There is also a ten–minute global timeout on all scripts being debugged. This means that even if you are performing user actions within the Debugger every two minutes, the Debugger will still timeout after ten minutes.

SuiteScript 2.1

A timeout can occur in two ways when using the 2.1 Script Debugger:

- There has been no user action within the debugger for five minutes. If you do not perform some user action in the debugger within five minutes, Chrome DevTools will disconnect and you will see this message:

  Clicking **Reconnect DevTools** does not resume your debug session. You must return to the Script Debugger tab and click the **X** button. Then click **Switch To Editor** and click **Debug Script** to start a new debug session. If you leave the Chrome DevTools tab open, it will be used for the new debugging session. If you close the Chrome DevTools tab, a new one will open when you start a new debugging session.
- You have reached the overall time limit for a debugger session. You set this time limit using the IDLE_SESSION_TIMEOUT_IN_MINUTES general preference, which sets the maximum time for a 2.1 Script Debugger session, regardless of whether there is user action or not. You can set the IDLE_SESSION_TIMEOUT_IN_MINUTES by going to Setup > General Preferences. Although you can enter a value larger than 20 minutes, a 2.1 Script Debugger session is limited to a maximum of 20 minutes. Note that this preference also sets the timeout for a NetSuite user session (see the help topic Setting General Account Preferences).

There are several errors thrown when a 2.1 Script Debugger timeout occurs:

- An error message "You have exceeded the maximum allowable idle time for debugging scripts. To debug another script, reload the script debugger page and start a new debugging session" will be thrown.
- DEBUGGER_SESSION_TIMEOUT error will be thrown if the timeout occurred based on the IDLE_SESSION_TIMEOUT_IN_MINUTES general preference value.
- DEBUGGER_IDLE_TIMEOUT error will be thrown if there has been no user action for five minutes.

If any of these timeouts occur, you can return to the Script Debugger page and click the **X** button to stop debugging. You can then click **Switch to Editor** to reload the script and start a new debugging session.

SuiteCloud Processors

Applies to: SuiteScript 2.x | SuiteCloud Developer

SuiteCloud Processors is the current system used to process scheduled scripts and map/reduce scripts. Before SuiteCloud Processors was introduced, scheduled scripts and map/reduce scripts were exclusively processed by scheduling queues. All scheduled script and map/reduce script jobs submitted to the same queue were processed on a FIFO (first in, first out) basis, based on the queue submission time stamp. This system had several limitations.

The scheduling queues did not provide automated load balancing or a way to prioritize specific jobs. Users with access to multiple queues (accounts with SuiteCloud Plus) were forced to manually determine the optimal configuration of jobs to queues. For the jobs that needed to be processed in a certain order, this method was useful. But it created unintended dependencies among many of the jobs submitted. If there was a delay in processing one job, a bottleneck would form. The result would be several jobs waiting in one queue when other queues were under utilized or not utilized at all.

SuiteCloud Processors resolves many of the limitations with scheduling queues. A scheduler now automatically determines the order in which jobs start to process. The scheduler uses algorithms that are based on user-defined priority levels, submission time, and user-defined preferences. The result is increased throughput, reduced wait times, and the elimination of most bottlenecks. In addition, SuiteCloud Processors requires less user intervention and enables map/reduce scripts and scheduled scripts to start sooner.

Note: Some features of SuiteCloud Processors are available only to accounts that have one or more SuiteCloud Plus licenses. For more information about SuiteCloud Plus, see the help topic SuiteCloud Plus Settings.

For additional information about SuiteCloud Processors, see:

- SuiteCloud Processors Terminology
- SuiteCloud Processors Basic Architecture
- SuiteCloud Processors Supported Task Types
- SuiteCloud Processors Processor Allotment Per Account
- SuiteCloud Processors Priority Levels
- SuiteCloud Processors Priority Elevation and Processor Reservation (Advanced Settings)
- Monitoring SuiteCloud Processors Performance

SuiteCloud Processors Terminology

Applies to: SuiteScript 2.x | SuiteCloud Developer

Term	Definition
Job	A job is a piece of work submitted to SuiteCloud Processors for processing. Each job is executed by a single processor.
Priority	A priority is a property of a job. The priority of submitted jobs determines the order in which the scheduler sends the jobs to the processor pool. Priorities are set on the deployment record or from the SuiteCloud Processors Priority Settings Page.
Processor	A processor is a virtual unit of processing power that executes a job. It is not distinguished as an individual physical entity, but as a single processing thread.

Term	Definition
Processor Pool	The processor pool represents the number of processors available to a specific account. For accounts without SuiteCloud Plus, the processor pool contains one processor. For accounts with SuiteCloud Plus, the processor pool contains multiple processors. For additional information, see the help topic SuiteCloud Plus Settings.
Queue	With SuiteCloud Processors, a queue is no longer a separate processing mechanism. On scheduled script deployments, the **Queue** field remains to accommodate deployments that rely on the FIFO (first in, first out) order imposed by an individual queue. However, all jobs that use queues are processed by the same processor pool that handles the jobs that do not use queues. All jobs compete with each other using the same common processing algorithm.
Scheduler	The scheduler determines the order in which jobs are sent to the processor pool. The scheduler uses algorithms that are based on user-defined priority levels, submission time, and user-defined preferences.
Task	A task is a script instance that is submitted for processing. Each task is handled by one or more jobs.

SuiteCloud Processors Basic Architecture

Applies to: SuiteScript 2.x | SuiteCloud Developer

SuiteCloud Processors currently supports the processing of scheduled scripts and map/reduce scripts. Each submitted scheduled script instance (task) is handled by one job. Each submitted map/reduce script task is handled by multiple jobs: one each for the getInput, shuffle, and summarize stages; and a minimum of one each for the map and reduce stages.

Scheduled script tasks and map/reduce script tasks are submitted for processing in one of three ways:

- By setting a one-time or recurring submission schedule from the script deployment record UI
- By selecting **Save and Execute** from the script deployment record UI to submit an on-demand instance of the script
- By using a SuiteScript API to submit an on-demand instance of the script

Important: To submit scheduled script tasks and map/reduce script tasks for processing, the role used to submit the tasks must have the following permissions:

- Documents and Files: View, Create, Edit, or Full
- SuiteScript: View, Create, Edit, or Full
- SuiteScript Scheduling: Full

For more information about access levels for permissions, see the help topic Access Levels for Permissions.

For additional information about submitting a script with the deployment record, see the following topics:

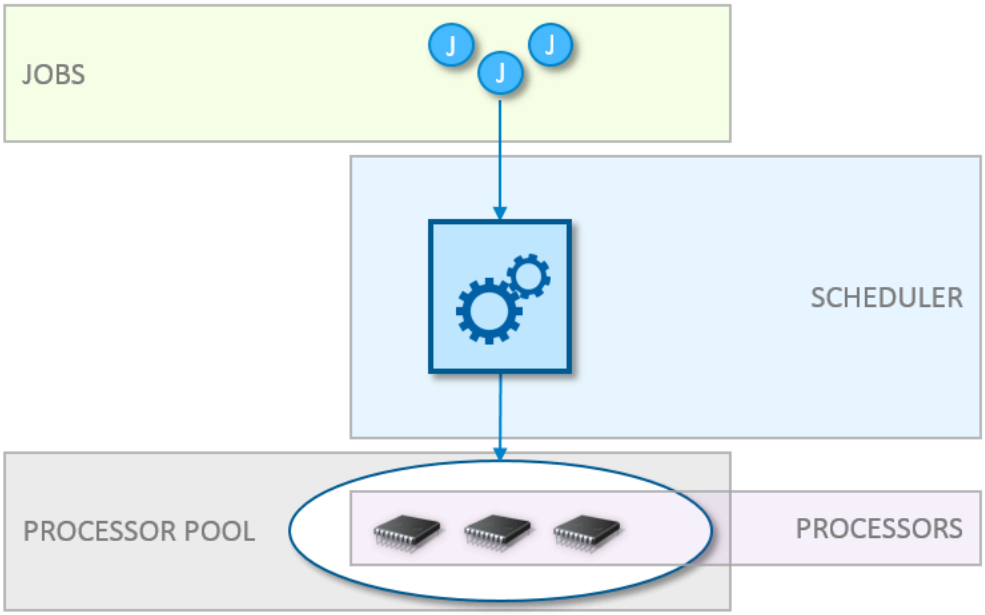
- SuiteScript 2.x Scheduled Script Type
- SuiteScript 2.x Map/Reduce Script Type

For additional information about submitting a script with a SuiteScript API, see the following topics:

- task.ScheduledScriptTask
- task.MapReduceScriptTask

The scheduler sends the resulting jobs to the processor pool in a particular order. This order is determined by the SuiteCloud Processors Priority Levels and the order of submission. Jobs with a higher priority are sent before jobs with a lower priority. Jobs with the same priority go to the processor pool in the order of submission.

SuiteCloud Processors includes advanced settings that can also impact the order in which jobs are sent to the processor pool. For additional information, see SuiteCloud Processors Priority Elevation and Processor Reservation (Advanced Settings).



SuiteCloud Processors Supported Task Types

Applies to: SuiteScript 2.x | SuiteCloud Developer

SuiteCloud Processors currently supports processing for two task types:

- SuiteScript 2.x Map/Reduce Script Type
- SuiteScript 2.x Scheduled Script Type

SuiteCloud Processors Impact on Map/Reduce Deployments

All map/reduce script deployments include a **Priority** field. For additional information, see SuiteCloud Processors Priority Levels.

For accounts with SuiteCloud Plus, the **Queues** multi-select field is replaced with the **Concurrency Limit** field. Instead of designating a specific queue, you set the maximum number of processors available to the deployment. This value equates to the number of jobs submitted for the map and reduce stages.

For map/reduce deployments created prior to the introduction of SuiteCloud Processors, the **Concurrency Limit** value is set by default. The default value is equivalent to the number of options selected on the **Queues** field. If **Queues** was previously set to 1, 3, 7, and 9, then **Concurrency Limit** is set to 4. If **Select All** was previously enabled, then **Concurrency Limit** is set to empty (the number of jobs initially created for the map and reduce stages is equivalent to the total number of processors in the processor pool).

For additional information, see the help topic SuiteScript 2.x Map/Reduce Script Type.

SuiteCloud Processors Impact on Scheduled Script Deployments

All scheduled script deployments include a **Priority** field. For additional information, see SuiteCloud Processors Priority Levels.

For all scheduled script deployments created **after** the introduction of SuiteCloud Processors, the **Queue** field is removed. These deployments cannot use queues.

For all scheduled script deployments created **prior** to the introduction of SuiteCloud Processors, the **Queue** field remains by default. This applies to accounts with and without SuiteCloud Plus. You have control over whether to stop using queues. The deployment record includes a **Remove Queue** option. After you select this option, the deployment no longer uses a queue **and cannot revert back to using a queue**.

The **Queue** field remains to accommodate deployments that rely on the FIFO order of processing imposed by an individual queue. However, all jobs that use queues are processed by the same processor pool that handles the jobs that do not use queues. All jobs compete with each other using the same common processing algorithm.

Note: For deployments that continue to use queues, all jobs assigned to the same queue should have the same priority. In most cases, you can keep the default (standard) priority of these jobs. However, in some cases, you may want to change these jobs to a higher or lower priority. One scenario is if you want to ensure that a specific queue always has a processor available. In that case, designate the jobs assigned to that queue as high priority. Alternatively, if you have a group of lower priority jobs, you can designate them as low priority and assign them to the same queue. That will ensure that only one is processed at a time.

Important: If your existing scheduled script deployments rely on implicit dependencies imposed by queues, you should update and test these scripts before you remove queues. Your scripts may be impacted if they rely on the sequence of FIFO (first in, first out).

One possible solution is to programmatically submit scripts in a certain order. To do this, use task.ScheduledScriptTask within the first script to submit the second script. This will ensure that the jobs are submitted to the processor pool in the correct order.

SuiteCloud Processors Processor Allotment Per Account

Applies to: SuiteScript 2.x | SuiteCloud Developer

Important: SuiteCloud Processors are used to execute (process) all scheduled script and map/reduce script instances. However, this topic does not apply to scheduled script deployments that continue to use queues. See the help topic SuiteCloud Plus Settings for more information.

SuiteCloud Processor Allotment

Service Tier	Maximum SuiteCloud Plus Licenses	Maximum SuiteCloud Processors
Standard	1	5

Service Tier	Maximum SuiteCloud Plus Licenses	Maximum SuiteCloud Processors
Premium	3	15
Enterprise	6	30
Ultimate	12	60

*The default number of SuiteCloud Processors available is 2 if you have not purchased any SuiteCloud Plus licenses.

For more information, see the help topics SuiteCloud Plus Settings and NetSuite Service Tiers.

Accounts Without a SuiteCloud Plus License

Accounts without a SuiteCloud Plus license now have access to two processors. The extra processor doubles the processing bandwidth for scheduled scripts and map/ reduce scripts.

If any of your scripts depend on implicit dependencies imposed by having one processor, you may be required to update your existing scripts for this feature. When an account has access to only one processor, all jobs are processed one at a time. If all jobs have the same priority, the order of processing is always first in, first out. Some scripts may depend on this behavior. With the addition of an extra processor, these scripts may no longer process in the correct order.

For example, if you have a script that pre-processes data for a second script, the first script must complete execution before the second script begins. With one processor, you can submit the scripts in the appropriate order and know that the order of processing is as expected. With two processors, there is a possibility that the second script submitted starts execution before the first script completes.

To handle the above scenario, perform the following steps:

1. Audit your scheduled scripts and map/reduce scripts. Look for scripts that depend on a specific order of execution.
2. Use the following SuiteScript APIs within the first script to programmatically submit the dependent script. This action ensures that the scripts are always executed in the correct order.

For a scheduled script, call task.ScheduledScriptTask and place the code at the end of the script.

For a map/reduce script, call task.MapReduceScriptTask and place the code at the end of the summarize stage.

SuiteCloud Processors Priority Levels

Applies to: SuiteScript 2.x | SuiteCloud Developer

With SuiteCloud Processors, script processing factors in submission time and priority level. This means that jobs with the highest priority are sent to the processor pool first. Jobs of lower priority are sent to the processor pool after all higher priority jobs are sent. The scheduler sends jobs with the same priority to the processor pool in the order of submission. For example, if jobs 1 and 4 are high priority, jobs 2 and 5 are standard priority, and job 3 is low priority, the scheduler sends the jobs to the processor pool in the following order:

- Job 1
- Job 4
- Job 2
- Job 5

- Job 3

By default, all jobs have a standard priority. When you change the default priority, you change when the scheduler sends the job to the processor pool. You can do this on the deployment record or on the SuiteCloud Processors Priority Settings Page. For examples that demonstrate how changing the priority can change the order of processing, see SuiteCloud Processors Priority Scheduling Examples.

The available priority options are:

- High: Use to mark critical jobs that require more immediate processing. The scheduler sends these jobs to the processor pool first.
- Standard: This is the default setting. It is considered to be a medium priority level. The scheduler sends these jobs to the processor pool if there are no high priority jobs waiting.
- Low: Use to mark jobs that can tolerate a longer wait time. The scheduler sends these jobs to the processor pool if there are no high or standard priority jobs waiting.

SuiteCloud Processors Priority Settings Page

Applies to: SuiteScript 2.x | SuiteCloud Developer

The Priority Settings page lists each scheduled script deployment and map/reduce script deployment created for your account. Each line item corresponds to one deployment record. To access the Priority Settings page, go to Customization > Scripting > Priority Settings.

The primary purpose of this page is to manage the SuiteCloud Processors Priority Levels for multiple deployments at one time. You can also use this page to remove queues for scheduled script deployments.

Buttons

Button	Description
Submit	Saves all changes made to the page
Mark All Queues for Removal	Places a check in the Remove Queue column for all scheduled script deployments that are still using queues. Queues are not shown as removed until you submit the page. If none of your scheduled scripts use queues, this button no longer appears.
Unmark All Queues for Removal	Removes all check marks currently in the Remove Queues column If none of your scheduled scripts use queues, this button no longer appears.
Reset Priorities	Resets all values in the Priority column to Standard (the default setting)

Filters

Filter	Description
Type	Filters for all map/reduce script deployments or all scheduled script deployments. Columns not applicable to the selected script type are hidden.
Status	Filters for the values listed in the Status column.
API Version	Filters for script deployments.
Script	Filters for all deployments of a specified script record
Page Size	Determines the number of line items shown on each result page.

Filter	Description
Show Undeployed	If enabled, the results include deployments where the Deployed option is disabled.

Columns

Column	Description
Internal ID	The internal ID for the script deployment record, as seen on the Script Deployments list page (for example, 345).
Edit | View	Links to the edit and view mode of the deployment record.
ID	The ID of the script deployment record (for example, customdeploy_testscript1).
Script	Corresponds to the Name field value on the script record associated with the deployment record.
API Version	Indicates the SuiteScript version.
Status	Indicates how and when a script can be submitted for processing. This value is set with the Status field on the deployment record. Possible values are: ■ **Testing**: Indicates that you can test the script in the SuiteScript Debugger. The script deployment can be submitted for processing by the script owner only. ■ **Scheduled**: Indicates that you can schedule a single or recurring instance of the script on the deployment record. You cannot submit an on demand instance of the script when it has this status. ■ **Not Scheduled**: Indicates that you can submit an on demand instance of the script with the **Save and Execute** button or a SuiteScript API. You can submit an on demand instance of the script only if there is no other unfinished instance of the same script.
Remove Queue	Only applicable to scheduled script deployments. Select this box to stop using queues on an existing scheduled script deployment. You can remove queues for multiple deployments at one time with this column. Queues are not shown as removed until you submit the page.
Type	Indicates whether the deployment is for a scheduled script or map/reduce script.
Queue	Only applicable to scheduled script deployments. Shows a value if the deployment is still using queues. After you remove queues, this value is empty.
Concurrency Limit	Only applicable to map/reduce script deployments on accounts that use SuiteCloud Plus. The maximum number of processors available to a map/reduce script deployment. You set this value on the map/reduce deployment record.
Priority	The priority setting for the deployment. For additional information, see SuiteCloud Processors Priority Levels.

SuiteCloud Processors Priority Scheduling Examples

Applies to: SuiteScript 2.x | SuiteCloud Developer

The two examples in this section, Default Priority Scheduling Example and Varying Priority Scheduling Example, each show three diagrams. The two sets of diagrams compare the same 18 jobs as they are handled by:

- Pre-2017.2 Scheduling Queues
- 2017.2 SuiteCloud Processors: All queues are removed

- 2017.2 SuiteCloud Processors: Some queues are removed; jobs 1 - 3, 15, and 18 still use queues

This table shows the duration of each job submitted. As demonstrated in the examples, the processing environment has no impact on the amount of time required to complete each job after processing starts.

Note: Each map/reduce stage is handled by a minimum of one job. Therefore, the duration listed for map/reduce jobs is per stage. For example, the reduce stage jobs 11 and 12 have a combined duration of 5. This means that the sum of the duration of jobs 11 and 12 is 5. When one of the jobs is canceled or not created, the combined duration becomes the duration of the remaining job from the pair.

Job	Duration
1	5
2	2
3	2
4	6
5 +6	2
7 + 8	1
9 + 10	5
11 + 12	5
13 + 14	2
15	3
16	3
17	3
18	2

Default Priority Scheduling Example

This example assumes default priority settings (standard priority) and default preferences. It also assumes that each job executes in full without yielding.

The map/reduce deployment in the first diagram has a **Queues** value of 2 and 3. The map/reduce deployment in the second and third diagrams has a **Concurrency Limit** value of 2.

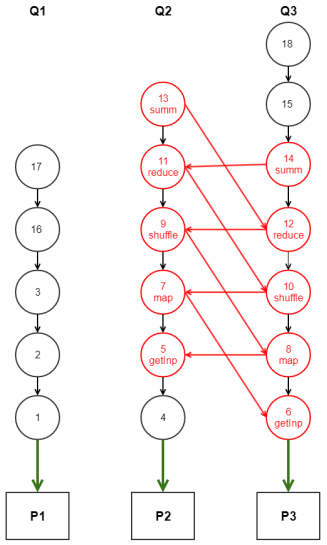
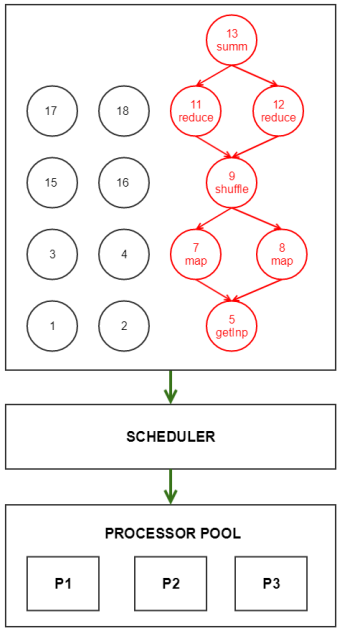
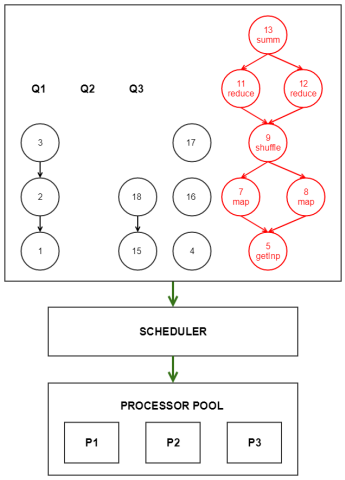
Note: With the introduction of SuiteCloud Processors, the **Queues** field is replaced with the **Concurrency Limit** field on map/reduce deployments.

Diagram Key

- The black circles are scheduled script jobs. Each scheduled script is processed by a single job.
- The red circles are map/reduce jobs. In this example, all of the map/reduce jobs belong to a single map/reduce task. The map/reduce jobs include a label that indicates the map/reduce stage being processed.
- The jobs in columns labeled Q1, Q2, and Q3 are for deployments that still use queues. The jobs in columns without a label are for deployments that no longer use queues.
- P1, P2, and P3 represent the processors that are available in the processor pool.

Note: Although processors are labeled for this example, they are not technically individual entities. Also, a real account would never have only three processors available to it. The minimum number of processors available to an account with SuiteCloud Plus is listed at SuiteCloud Plus Settings.

- The numbers within each circle represent the time stamp on the job submission, with 1 being the first job submitted. The example also uses this number as a unique identifier for each job.
- Arrows represent dependencies. For example, in the Scheduling Queues diagram, job 2 cannot start until job 1 is complete.
  - Black arrows represent queue dependencies. Jobs within a queue must be processed in the order they were submitted.
  - Red arrows represent dependencies imposed by the map/reduce algorithm.
  - Green arrows indicate access to processors.

Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
		

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
101	Jobs **1**, **4**, and **6** start.	Jobs **1**, **2**, and **3** start.	Jobs **1**, **4**, and **5** start.
102	Job **6** cancels job **5**. The getInput stage cannot be processed by multiple jobs, so the first getInput job to start (job **6**) automatically cancels all others in the queues.	—	—
103	Job **6** completes.	Jobs **2** and **3** complete. Jobs **4** and **5** start.	Job **5** completes. There is no job **6** with SuiteCloud Processors. In the Scheduling Queues

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
	The next job in Q3, job **8**, starts.		example, jobs **5** and **6** are both getInputstage jobs. The getInput stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple getInput stage jobs are no longer needed. Job **7** starts.
104	Job **8** completes. The map stage can be processed by multiple jobs. But since the other map stage job, job **7**, is unable to start (the job utilizing its processor, job **4**, is not yet complete), job **8** completes the entire map stage. Job **7** is no longer needed, so it is canceled. The next job in Q3, job **10**, starts.	—	Job **7** completes. Since job **7** completes the entire map stage, job 8 is no longer needed. Job **8** is canceled. Job **9** starts.
105	Job **10** cancels job **9**. The shuffle stage cannot be processed by multiple jobs, so the first shuffle job to start (job **10**) automatically cancels all others in the queues.	Job **5** completes. There is no job **6** with SuiteCloud Processors. In the Scheduling Queues example, jobs **5** and **6** are both getInputstage jobs. The getInput stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple getInput stage jobs are no longer needed. Job **7** starts.	—
106	Job **1** completes. The next job in Q1, job **2**, starts.	Job **7** completes. Since job **7** completes the entire map stage, job **8** is no longer needed. Job **8** is canceled. Job **1** completes. Jobs **9** and **15** start. Jobs **11 - 13** are dependent on job **9**. There is no job **10** or job **14** with SuiteCloud Processors. In the Scheduling Queues example, jobs **9** and **10** are both shuffle stage jobs and jobs **13** and **14** are both summarize stage jobs. The shuffle and summarize stages cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple shuffle and summarize stage jobs are no longer needed.	Job **1** completes. The next job in Q1, job **2**, starts.
107	Job **4** completes. Q2 is blocked. The next job in Q2, job **11**, cannot start. It is dependent on job **10**, and job **10** is not complete.	—	Job **4** completes. The first job in Q3, Job **15**, starts.

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
108	Job **2** completes. The next job in Q1, job **3**, starts.	—	Job **2** completes. The next job in Q1, job **3**, starts.
109	Job **10** completes. Job **11** can now start and Q2 is no longer blocked. The next job in Q3, job **12**, starts. These are both reduce stage jobs. The reduce stage can be processed by multiple jobs.	Jobs **4** and **15** complete. Jobs **16** and **17** start.	Job **9** completes. There is no job **10** with SuiteCloud Processors. In the Scheduling Queues example, jobs **9** and **10** are both shuffle stage jobs. The shuffle stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple shuffle stage jobs are no longer needed. Job **11** starts.
110	Job **3** completes. The next job in Q1, job **16**, starts.	—	Jobs **3** and **15** complete. There are no additional jobs to process in Q1. Jobs **12** and **16** start.
111	Job **11** completes. Q2 is blocked. The next job in Q2, job **13**, cannot start. It is dependent on job **12**, and job **12** is not complete.	Job **9** completes. Job **11** starts.	—
112	Job **12** completes. Job **13** can now start and Q2 is no longer blocked. The next job in Q3, job **14**, can also start. But these are both summarize stage jobs, and the summarize stage cannot be processed by multiple jobs. Job **13** starts first, so job **14** is canceled. Since job **14** is canceled, the next job in Q3, job **15**, starts	Jobs **16** and **17** complete. Jobs **12** and **18** start.	Jobs **11** and **12** complete. Jobs **13** and **17** start.
113	Job **16** completes. The next job in Q1, job **17**, starts.	—	Job **16** completes. The next job in Q3, Job **18**, starts.
114	Job **13** completes. There are no additional jobs to process in Q2.	Jobs **11**, **12**, and **18** complete. Job **13** starts	Job **13** completes. There is no job **14** with SuiteCloud Processors. In the Scheduling Queues example, jobs **13** and **14** are both summarize stage jobs. The summarize stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
			summarize stage jobs are no longer needed.
115	Job **15** completes. The next job in Q3, job **18**, starts.	—	Jobs **17** and **18** complete. There are no additional jobs to process.
116	Job **17** completes. There are no additional jobs to process in Q1.	Job **13** completes. There are no additional jobs to process.	—
117	Job **18** completes. There are no additional jobs to process in Q3.	—	—

Varying Priority Scheduling Example

This example assumes default preferences. It also assumes that each job executes in full without yielding.

The map/reduce deployment in the first diagram has a **Queues** value of 2 and 3. The map/reduce deployment in the second and third diagrams has a **Concurrency Limit** value of 2.

Note: With the introduction of SuiteCloud Processors, the **Queues** field is replaced with the **Concurrency Limit** field on map/reduce deployments.

The second and third diagrams vary from Default Priority Scheduling Example as follows:

- In the second diagram, jobs 15 and 18 are high priority. Job 4 is low priority. The remaining jobs are standard priority.
- In the third diagram, job 4 is low priority. Jobs 15 and 18 in the third diagram are standard priority.

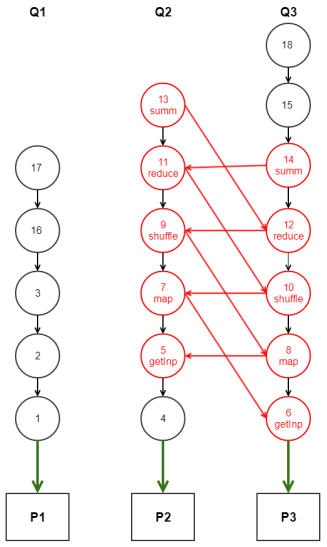
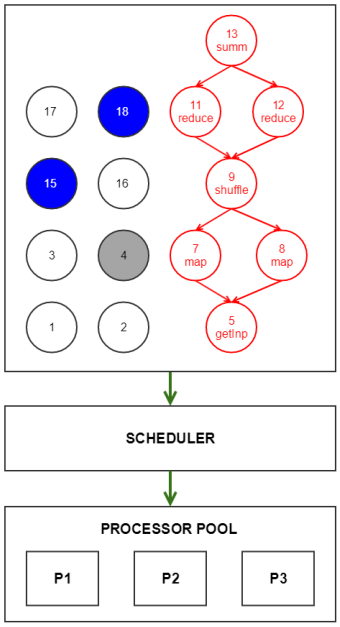
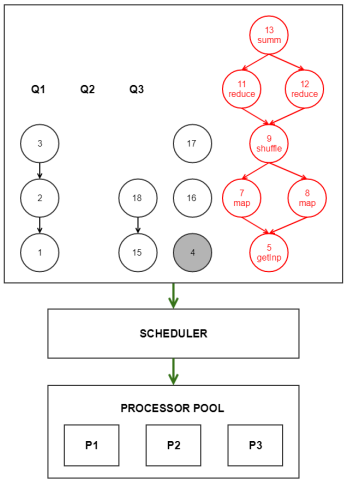
Diagram Key

- The black circles are scheduled script jobs. Each scheduled script is processed by a single job.
- The red circles are map/reduce jobs. In this example, all of the map/reduce jobs belong to a single map/reduce task. The map/reduce jobs include a label that indicates the map/reduce stage being processed.
- **The circles with blue fill are high priority. The circles with gray fill are low priority. The circles with white fill are standard priority.**
- The jobs in columns labeled Q1, Q2, and Q3 are for deployments that still use queues. The jobs in columns without a label are for deployments that no longer use queues.
- P1, P2, and P3 represent the processors that are available in the processor pool.

  Note: The processors are labeled for this example. Although technically, processors are not distinguished as individual entities. Also, a real account would never have only three processors available to it. The minimum number of processors available to an account with SuiteCloud Plus is listed at SuiteCloud Plus Settings.

- The numbers within each circle represent the time stamp on the job submission, with 1 being the first job submitted. The example also uses this number as a unique identifier for each job.

- Arrows represent dependencies. For example, in the Scheduling Queues diagram, job 2 cannot start until job 1 is complete.
  - Black arrows represent queue dependencies. Jobs within a queue must be processed in the order they were submitted.
  - Red arrows represent dependencies imposed by the map/reduce algorithm.
  - Green arrows indicate access to processors.

Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
		

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
101	Jobs **1**, **4**, and **6** start.	Jobs **15**, **18**, and **1** start. Jobs **15** and **18** are designated as high priority so they are assigned to processors first (in the order of submission). Job **1** is the first standard priority job submitted so it is the next job assigned to a processer.	Jobs **1**, **5**, and **15** start. Job **4** is designated low priority, so it is processed after all other available standard priority jobs. All other standard priority jobs submitted before job **15** have dependencies that are blocking them.
102	Job **6** cancels job **5**. The getInput stage cannot be processed by multiple jobs, so the first getInput job to start (job **6**) automatically cancels all others in the queues.	—	—
103	Job **6** completes. The next job in Q3, job **8**, starts.	Job **18** completes. Job **2** starts.	Job **5** completes. There is no job **6** with SuiteCloud Processors. In the Scheduling Queues example, jobs **5** and **6** are both getInputstage jobs. The getInput stage cannot be processed by multiple jobs. Since

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
			this map/reduce deployment is not using queues, multiple getInput stage jobs are no longer needed. Job **7** starts.
104	Job **8** completes. The map stage can be processed by multiple jobs. But since the other map stage job, job **7**, is unable to start (the job utilizing its processor, job **4**, is not yet complete), job **8** completes the entire map stage. Job **7** is no longer needed, so it is canceled. The next job in Q3, job **10**, starts.	Job **15** completes. Job **3** starts.	Jobs **7** and **15** complete. Since job **7** completes the entire map stage, job 8 is no longer needed. Job **8** is canceled. Jobs **9** and **16** start.
105	Job **10** cancels job **9**. The shuffle stage cannot be processed by multiple jobs, so the first shuffle job to start (job **10**) automatically cancels all others in the queues.	Job **2** completes. Job **5** starts. Job **4** is designated low priority, so it is processed after all other available standard priority jobs.	—
106	Job **1** completes. The next job in Q1, job **2**, starts.	Jobs **1** and **3** complete. Jobs **16** and **17** start.	Job **1** completes. The next job in Q1, job **2**, starts.
107	Job **4** completes. Q2 is blocked. The next job in Q2, job **11**, cannot start. It is dependent on job **10**, and job **10** is not complete.	Job **5** completes. There is no job **6** with SuiteCloud Processors. In the Scheduling Queues example, jobs **5** and **6** are both getInputstage jobs. The getInput stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple getInput stage jobs are no longer needed. Job **7** starts.	Job **16** completes. Job **17** starts.
108	Job **2** completes. The next job in Q1, job **3**, starts.	Job **7** completes. Since job **7** completes the entire map stage, job 8 is no longer needed. Job **8** is canceled. Job **9** starts.	Job **2** completes. The next job in Q1, job **3**, starts.
109	Job **10** completes. Job **11** can now start and Q2 is no longer blocked. The next job in Q3, job **12**, starts. These are both reduce stage jobs. The reduce stage can be processed by multiple jobs.	Jobs **16** and **17** complete. Job **4** starts. Job **4** is designated as low priority, but all the remaining standard priority jobs are dependent on job **9**.	Job **9** completes. There is no job **10** with SuiteCloud Processors. In the Scheduling Queues example, jobs **9** and **10** are both shuffle stage jobs. The shuffle stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple shuffle stage jobs are no longer needed. Job **11** starts.
110	Job **3** completes.	—	Jobs **3** and **17** complete.

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
	The next job in Q1, job **16**, starts.		There are no additional jobs to process in Q1. Job **12** starts. The next job in Q3, Job **18**, starts.
111	Job **11** completes. Q2 is blocked. The next job in Q2, job **13**, cannot start. It is dependent on job **12**, and job **12** is not complete.	—	—
112	Job **12** completes. Job **13** can now start and Q2 is no longer blocked. The next job in Q3, job **14**, can also start. But these are both summarize stage jobs, and the summarize stage cannot be processed by multiple jobs. Job **13** starts first, so job **14** is canceled. Since job **14** is canceled, the next job in Q3, job **15**, starts	—	Jobs **11**, **12** and **18** complete. There are no additional jobs to process in Q3. Jobs **13** and **4** start.
113	Job **16** completes. The next job in Q1, job **17**, starts.	Job **9** completes. There is no job **10** with SuiteCloud Processors. In the Scheduling Queues example, jobs **9** and **10** are both shuffle stage jobs. The shuffle stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple shuffle jobs are no longer needed. Jobs **11** and **12** start.	—
114	Job **13** completes. There are no additional jobs to process in Q2.	—	Job **13** completes. There is no job **14** with SuiteCloud Processors. In the Scheduling Queues example, jobs **13** and **14** are both summarize stage jobs. The summarize stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple summarize stage jobs are no longer needed.
115	Job **15** completes. The next job in Q3, job **18**, starts.	Jobs **4** and **12** complete.	—
116	Job **17** completes. There are no additional jobs to process in Q1.	Job **11** completes. Job **13** starts.	—
117	Job **18** completes.	—	—

Time Slot	Scheduling Queues Only	SuiteCloud Processors – All Queues Removed	SuiteCloud Processors – Some Queues Removed
	There are no additional jobs to process in Q3.		
118	—	Job **13** completes. There is no job **14** with SuiteCloud Processors. In the Scheduling Queues example, jobs **13** and **14** are both summarize stage jobs. The summarize stage cannot be processed by multiple jobs. Since this map/reduce deployment is not using queues, multiple summarize stage jobs are no longer needed. There are no additional jobs to process.	Job **4** completes. There are no additional jobs to process.

SuiteCloud Processors Priority Elevation and Processor Reservation (Advanced Settings)

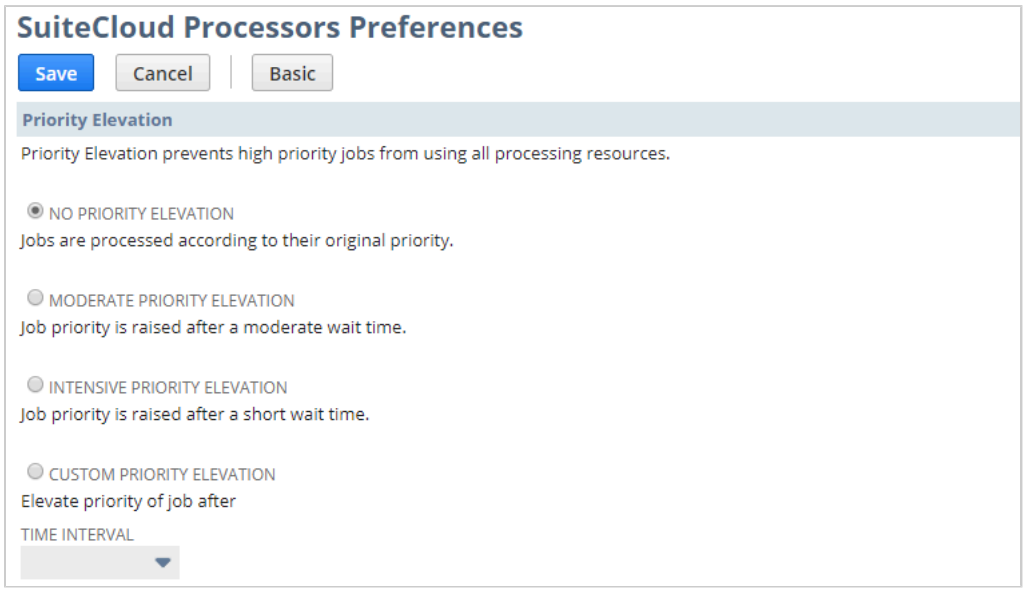
Applies to: SuiteScript 2.x | SuiteCloud Developer

Priority Elevation

Priority elevation enables you to automatically increase the priority of each low or standard priority job after a specific time interval. The time interval starts when the job is submitted. You do not need to use these settings in most cases. Because of this, priority elevation is disabled by default. However, you may need to use them if lower priority jobs experience excessive wait times.

Note: Priority elevation only impacts lower priority jobs with a wait time greater than the time interval indicated. If a lower priority job is sent to the processor pool before the time interval is over, that job is processed with its original priority.

To access the priority elevation settings, go to Setup > Preferences > SuiteCloud Processors.



There are three predetermined settings listed:

- **No Priority Elevation**: The default setting
- **Moderate Priority Elevation**:
  - If a low priority job is still waiting after four hours, it is elevated to standard priority
  - If a standard priority job is still waiting after four hours, it is elevated to high priority

  With this option, if priority elevation is applicable, the system elevates low priority jobs to high priority jobs after eight hours. Specifically, the jobs in this scenario are elevated to standard priority after four hours and then elevated to high priority after another four hours.
- **Intensive Priority Elevation**:
  - If a low priority job is still waiting after one hour, it is elevated to standard priority
  - If a standard priority job is still waiting after one hour, it is elevated to high priority

  With this option, if priority elevation is applicable, the system elevates low priority jobs to high priority jobs after two hours. Specifically, the jobs in this scenario are elevated to standard priority after one hour and then elevated to high priority after another hour.
- **Custom Priority Elevation**: This option enables you to specify a custom time interval for priority elevation from the **Time Interval** dropdown list.

The **Time Interval** field specifies the time interval set for priority elevation. When **Custom Priority Elevation** is selected, this field is enabled for editing. Otherwise, the field displays a value that corresponds with the option selected, but it cannot be edited.

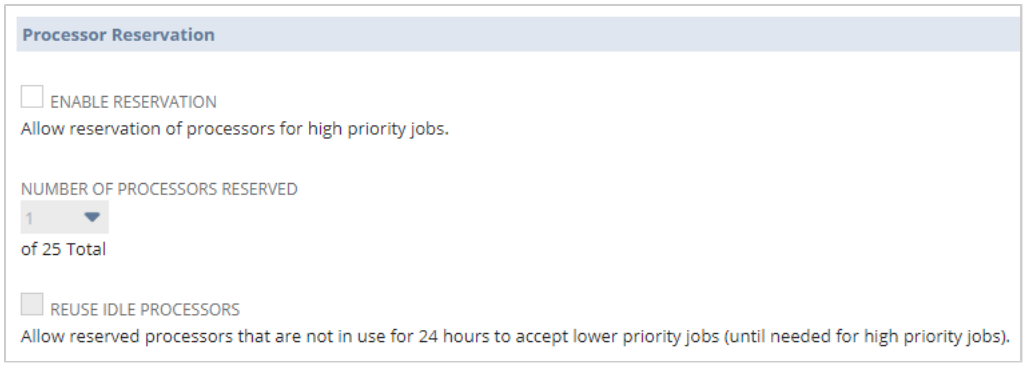
Note: Click **Advanced** at the top of the page to access **Custom Priority Elevation** and **Time Interval**.

Processor Reservation

Processor reservation enables you to reserve a specified number of processors for high priority jobs. You do not need to use these settings in most cases. Because of this, processor reservation is disabled by default.

Important: Processor reservation is available for SuiteCloud Plus accounts only.

To access the priority elevation settings, go to Setup > Preferences > SuiteCloud Processors. Click **Advanced** at the top of the page.



When you select **Enable Reservation**, you can reserve all but one of your available processors from the **Number of Processors Reserved** dropdown list. If a high priority job is submitted, it is sent to the processor pool if there is at least one processor available. If a standard or low priority job is submitted, it is sent to the processor pool only if there are more processors available than the number reserved. For example, you have 10 processors reserved out of 25 total processors. A standard or low priority job is sent to the processor pool if there are at least 11 processors available. If there are 10 processors or fewer available, the lower priority job must wait.

Important: Changes to the Number of Processors Reserved apply to all jobs that have not yet started. This can have an immediate impact on map/reduce scripts since each stage is processed by a minimum of one job. If there is a high priority map/reduce script instance (task) executing and this setting is changed, the new value is applied to all jobs for this task that have not yet started. This includes jobs that may be created from yielding.

Processor reservation decreases the number of processors available for standard and low priority jobs. Therefore, it can reduce the throughput of these jobs. The **Reuse Idle Processors** setting temporarily releases reserved processors that have not been used in the past 24 hours. This increases the number of processors for lower priority jobs.

When **Reuse Idle Processors** is enabled, it initiates an hourly recurring audit. The system uses the data collected to determine whether to release reserved processors. After reserved processors are released, the audit data is also used to determine whether the system needs to increase reserved processors.

Important: If the system decreases or increases the number of reserved processors, additional decreases are not made for 24 hours. However, additional increases (up to the selected limit) can still be made after each hourly audit. This process continues if **Reuse Idle Processors** is enabled.

The system analyzes the following data points during this process:

- **a** = total number of processors available
- **b** = the value you set for **Number of Processors Reserved** (the maximum number of reserved processors; this number does not change)
- **c** = maximum number of jobs concurrently processed in the last 24 hours
- **d** = maximum number of high priority jobs that concurrently waited more than a minute in the last hour

- **e** = current number of reserved processors (this number can change if **Reuse Idle Processors** is enabled)

If	Then
c is less than **a** This means that some reserved processors were not used.	The number of processors available to lower priority jobs is increased by **a – c** (up to the value of **a**).
e is less than **b AND d** is more than **0**	The number of reserved processors is increased by **d** (up to the value of **b**) The system cannot increase the number of reserved processors over the limit set for **Number of Processors Reserved**. In other words, **e** cannot be greater than **b**.

SuiteScript Monitoring, Auditing, and Logging

Applies to: SuiteScript 2.x | SuiteCloud Developer

SuiteScript execution can be monitored and audited using script logs. Script execution details can be viewed on the Script page, the Script Deployment page, and the SuiteScript Debugger. A list of all records that have a user event script or a client script associated with a record can be viewed on the Scripted Record page. You can also set several runtime options to specify when a script is executed. And, the Application Performance Management (APM) SuiteApp can be used to view and manage the performance of NetSuite customizations and business critical operations. See the following help topics for more information:

- Using the Script Execution Log Tab
- Viewing a List of Script Execution Logs
- The Scripted Records Page
- SuiteScript Monitoring with the Application Performance Management (APM)
- Setting Runtime Options
- Governance on Script Logging
- Using the Context Filtering Tab
- Reviewing Outbound HTTPS and SFTP Requests

Using the Script Execution Log Tab

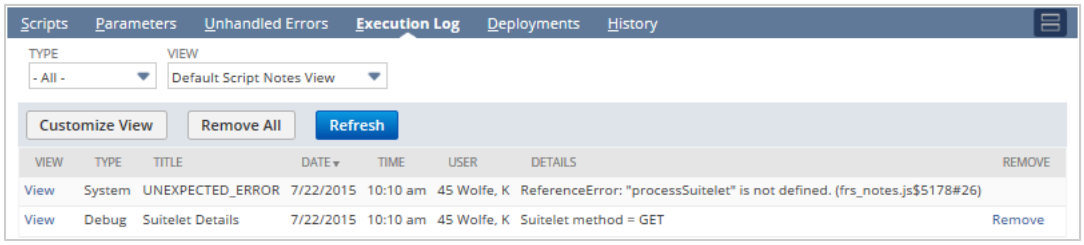
Applies to: SuiteScript 2.x | SuiteCloud Developer

Script execution details are logged on the Execution Log tab included on the Script page, Script Deployment page, and SuiteScript Debugger. These logs are searchable, and you can customize views to find specific logs.

The Execution Log tab displays all dates and times in the local time zone set in the user preferences.

Important: When using the SuiteScript Debugger to debug a script, all logging details appear on the Execution Log tab in the Debugger. To have logging details appear on the Execution Log tab of the Script Deployment page, you must deploy the script first.

The following figure shows two types of execution logs for a Suitelet.



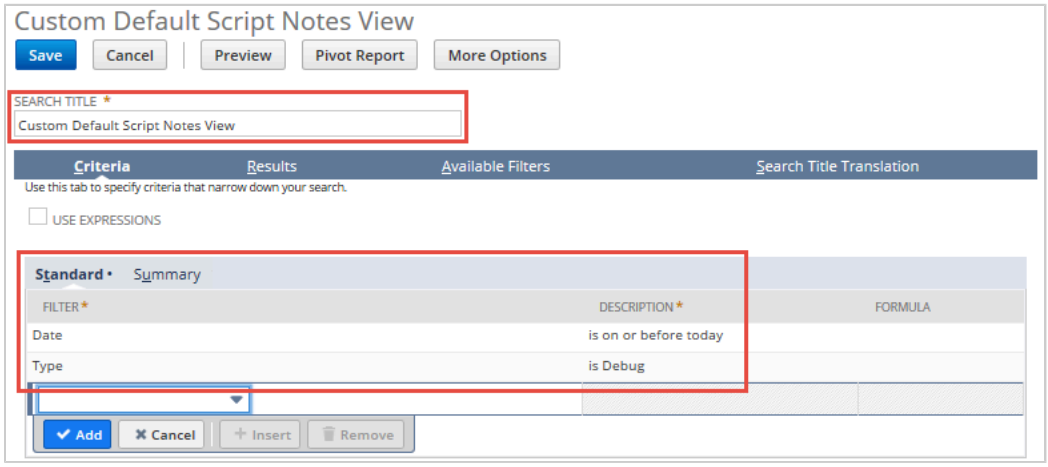
The first one is an unexpected error that is generated because a method was not defined in a Suitelet script. The second one is a user-generated execution log that is generated by the following line(s) in this Suitelet code using the SuiteScript 2.0 N/log Module:

```
1 | log.debug({
```

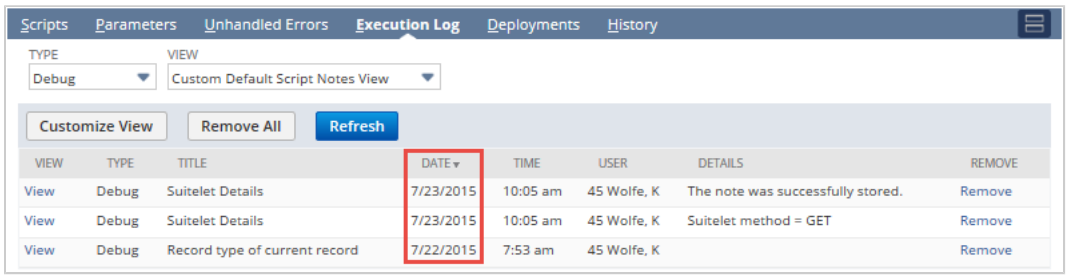
```
    title: 'Suitelet Details',
    details: 'Suitelet method = ' + request.getMethod()
});
```

By default, the View list on the Execution Log tab is set to Default Script Notes View. This default view shows all log types from the current day's script executions.

To view script executions for days other than the current day, click the Customize View button. Specify search criteria such as script execution dates, details, names, and script types, and then name your custom view in the Search Title field, as shown here:



The following figure shows a customized view of the execution log. The new view type has been selected from the View list. This view shows log details for days other than the current day.



For more information about viewing script execution logs, see Viewing a List of Script Execution Logs.

Important: The capacity for script execution logs on the Execution Log tab is shared by customers on the same database. For further protection against excessive logging, script execution logs are governed by a total storage limit on each instance of the NetSuite database. On each NetSuite server, if the database table that stores logs reaches this limit, all logs (across all customers on that server) are purged. This means that you may have logs up to 30 days if the volume is low, but you may have logs of less than a week if the log volume is extremely large. Use the Script Execution Log page for logs up to 30 days, regardless of volume. For more information about governance of script logs, see the help topic Governance on Script Logging

Viewing a List of Script Execution Logs

Applies to: SuiteScript 2.x | SuiteCloud Developer

The Script Execution Log page shows log details across all scripts for 30 days. This page can be accessed through Customization > Scripting > Script Execution Logs.

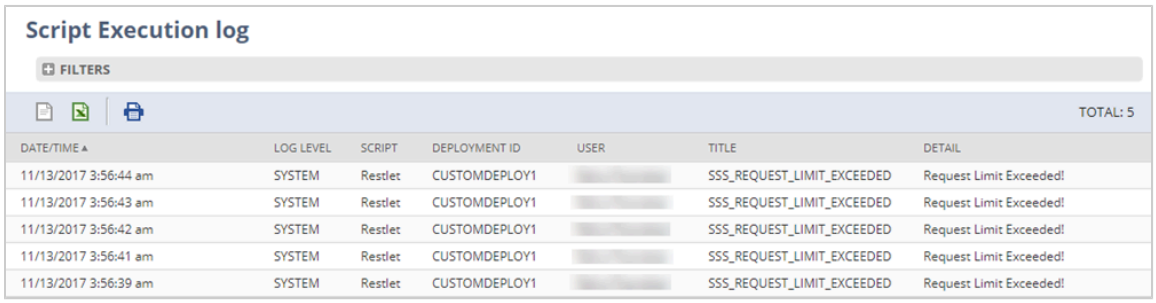
The Script Execution Log page displays all dates and times in the local time zone set in the user preferences.

On this page, you can perform the following tasks:

- Search for specific logs using filter options, such as log level, execution date range, and script name. You can view other logs by using filter options.
- Download the list as a CSV file or an Excel spreadsheet. Downloads may be limited to 10,000 entries.
- Print the log list.

Note: These logs are not moved with your NetSuite account to the new data center built on Oracle Cloud Infrastructure (OCI). For the first 29 days after the move, the displayed values will be calculated using data stored since the date of the move, rather than from the last 30 days.

The following figure shows a script execution log list that has been filtered to show scripts that violated the web services and RESTlet account concurrency governance.



The Scripted Records Page

Applies to: SuiteScript 2.x | SuiteCloud Developer

You can view a list of all records that have a user event script or a client script associated with a record on the Scripted Records page. By default, only records that have at least one associated script appear in the list.

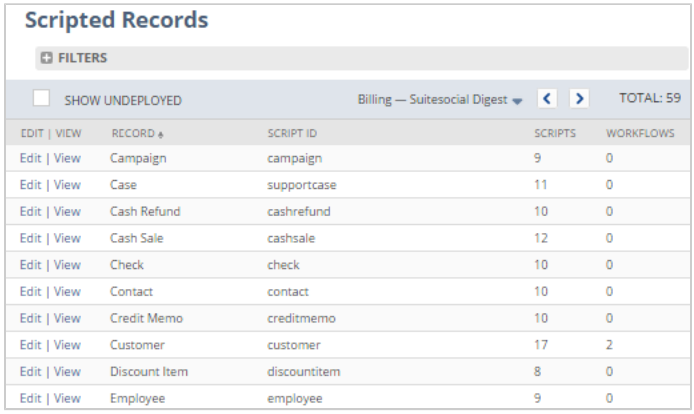
Tip: To view a list of all records in your account, click the Show Undeployed box. You can also use the Script filter list to show only records associated with specific scripts.

The Scripted Records page is helpful for troubleshooting problematic scripts and workflows. You can use the page to:

- access a specific record type to specify the execution order of scripts associated with each record of that type
- edit script deployment statuses
- deploy or inactivate script deployments
- change workflow statuses
- set workflow initiation details

In order of execution, scripts which are not localized takes precedence over those who are localized. For information about Localization Context, see Localization Context.

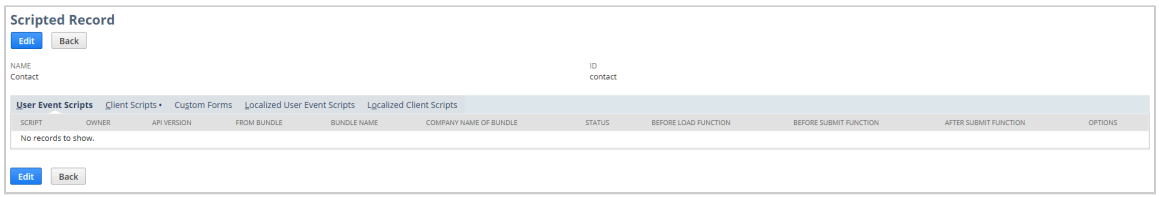
To access the Scripted Records page, go to Customization > Scripting > Scripted Records.



Each specific record is shown on the Scripted Record page. To access a specific scripted record:

- Click **View** to view information about the scripts associated with a record type, or
- Click **Edit** to change a record's script deployment order, or to deploy or undeploy scripts for that record type.

The Scripted Record page includes these subtabs: User Event Scripts, Client Scripts, Custom Forms, Localized User Event Scripts, Localized Client Scripts, and Workflows (if enabled).



User Event Scripts Subtab

Use the **User Event Scripts** subtab to:

- View information about the user event scripts for a record type:
  - The **Script** column lists the script's name. Click the script's name to view its Script Deployment record. See the help topic Script Deployment.
  - The **Owner** column lists the script's owner.
  - The **API Version** column lists the script's API version.
  - The **From Bundle** column lists the number of the bundle the script is included with. Click the bundle number to view information about the bundle.
  - The **Bundle Name** column lists the name of the bundle that the script is included with. Click the bundle name to view information about the bundle.
  - The **Company Name of Bundle** column lists the name of the bundle that the company uses to identify the bundle.
  - The **Status** column lists the release status of the script, either Testing or Released. See the help topic Setting Script Deployment Status.

- The **Before Load Function** column lists the names of the functions that are set to execute on the `beforeLoad` user event entry point.
- The **Before Submit Function** column lists the names of the functions that are set to execute on the `beforeSubmit` user event entry point.
- The **After Submit Function** column lists the names of the functions that are set to execute on the `afterSubmit` user event entry point.
- The **Options** column lists any options in the script and their setting.

Note: For information about user event script entry points, see the help topic SuiteScript 2.x User Event Script Entry Points and API.

- Change the script execution order of user event scripts.

Note: Scripts execute by function order first, then by the order listed on the Scripted Records page. Individual execution sequences for `beforeLoad`, `beforeSubmit`, and `afterSubmit` cannot be defined separately. When `afterSubmit` functions are executed, those with a `beforeSubmit` function will always go first, and after that, the execution sequence defined by the customer will be respected. For example, deploy four user event scripts on any record:

- In the first script (A), define `beforeLoad` and `afterSubmit` entry points only
- In the second script (B), define `beforeLoad` and `beforeSubmit` entry points only
- In the third script (C), define `afterSubmit` entry point only
- In the fourth script (D), define `beforeSubmit` and `afterSubmit` entry points only

The entry points will be executed in this order:

- Script A `beforeLoad`
- Script B `beforeLoad`
- Script B `beforeSubmit`
- Script D `beforeSubmit`
- Script D `afterSubmit`
- Script A `afterSubmit`
- Script C `afterSubmit`

Notice, that the `afterSubmit` function of Script D is executed before the `afterSubmit` functions of Script A and Script C, even if Script D is listed last in the execution order on the User Events tab.

Note: When there is a workflow defined for the same record as a user event script, and both include `afterSubmit` events, the user event script is executed in its entirety before the workflow's `afterSubmit` event.

- Change a script's deployment status from Testing to Released.

- Deploy a script by clicking the **Deployed** box.

Tip: Deploying a large number of user event scripts of the same event type can impact performance. Avoid deploying more than 10 scripts of the same event type.

Client Scripts Subtab

A maximum of 10 localized/non-localized client scripts are supported.

Use the **Client Scripts** subtab to:

- Change the script execution order of global client scripts. For example, have the third script execute first by moving the script to the top of the list.
- Change a script's deployment status from Testing to Released.
- Deploy a script by checking the **Deployed** box.
- View information about the client scripts for a record type:
  - The **Script** column lists the script's name. Click the script's name to view its Script Deployment record. See the help topic Script Deployment.
  - The **Owner** column lists the script's owner.
  - The **API Version** column lists the script's API version.
  - The **From Bundle** column lists the number of the bundle the script is included with. Click the bundle number to view information about the bundle.
  - The **Bundle Name** column lists the name of the bundle that the script is included with. Click the bundle name to view information about the bundle.
  - The **Company Name of Bundle** column lists the name of the bundle that the company uses to identify the bundle.
  - The **Status** column lists the release status of the script, either Testing or Released. See the help topic Setting Script Deployment Status.
  - The **Page Init Function** column lists the names of the functions that are set to execute on the `pageInit` client script entry point.
  - The **Save Record Function** column lists the names of the functions that are set to execute on the `saveRecord` client script entry point.

    The **Field Changed Function** column lists the names of the functions that are set to execute on the `fieldChanged` client script entry point.
  - The **Validate Line Function** column lists the names of the functions that are set to execute on the `validateLine` client script entry point.

Note: For information about client script entry points, see the help topic SuiteScript 2.x Client Script Entry Points and API.

Custom Forms Subtab

Use the **Custom Forms** subtab to:

- View information about the custom forms for a record type:
  - The **Form** column lists the custom form's name.
  - The **From Bundle** column lists the number of the bundle the form is included with. Click the bundle number to view information about the bundle.

- The **Bundle Name** column lists the name of the bundle that the form is included with. Click the bundle name to view information about the bundle.
- The **Company Name of Bundle** column lists the name of the bundle that the company uses to identify the bundle.
- The **Script** column lists the script's name. Click the script's name to view its Script Deployment record. See the help topic Script Deployment.
- The **Page Init Function** column lists the names of functions that are set to execute on the `pageInit` client script entry point.
- The **Save Record Function** column lists the names of the functions that are set to execute on the `saveRecord` client script entry point.
- The **Field Changed Function** column lists the names of the functions that are set to execute on the `fieldChanged` client script entry point.
- The **Validate Line Function** column lists the names of the functions that are set to execute on the `validateLine` client script entry point.

Workflows Subtab

Use the **Workflows** subtab to select a workflow to:

- Change the status of the workflow. Use the **Status** list to set the status to Testing, Not Running, or Released.
- Set the initiation trigger type: All, Before Record Load, Before Record Submit, After Record Submit. Use the **Trigger Type** list to set the type of trigger on which the workflow is to initiate. Note that this is not available for workflows that are locked by a bundle.
- Specify whether the workflow should initiate on create events by checking the **On Create** box. Note that this is not available for workflows that are locked by a bundle. See the help topic Initiating a Workflow on an Event.
- Specify whether the workflow should initiate on view or update events by checking the **On View or Update** box. Note that this is not available for workflows that are locked by a bundle. See the help topic Initiating a Workflow on an Event.
- View information about the workflows for a record type:
  - The **Workflow** column lists the workflow's name. Click the name to view the workflow in the SuiteFlow UI. See the help topic Workflow Manager Interface.
  - The **Internal ID** column lists the workflow's internal ID. Click the ID to view the workflow in the SuiteFlow UI. See the help topic Workflow Manager Interface.
  - The **Description** column lists a description of the workflow, if available.
  - The **From Bundle** column lists the number of the bundle the form is included with. Click the bundle number to view information about the bundle.
  - The **Bundle Name** column lists the name of the bundle that the form is included with. Click the bundle name to view information about the bundle.
  - The **Company Name of Bundle** column lists the name of the bundle that the company uses to identify the bundle.
  - The **Owner** column lists the workflow's owner.
  - The **Status** column lists the release status of the script: Testing, Not Running, or Released. See the help topic Release Status.
  - The **Trigger Type** column lists the client event on which the workflow is set to initiate: All, Before Record Load, Before Record Submit, After Record Submit. See the help topic Client Triggers Reference.

- The **On Create** column lists whether the workflow initiates on record creation events. See the help topic Initiating a Workflow on an Event.
- The **On View or Update** column lists whether the workflow initiates on record view or update events. See the help topic Initiating a Workflow on an Event.

For information about SuiteFlow workflows, see the help topic SuiteFlow Overview. Also note that the SuiteFlow feature must be enabled to use workflows. See the help topic Enabling SuiteFlow.

Localized User Event Scripts Subtab

Use the **Localized User Event Scripts** subtab to:

- Change the script execution order of localized user event scripts. For example, have the third script execute first by moving the script to the top of the list.
- Change a script's deployment status from Testing to Released.
- Deploy a script by clicking the **Deployed** box.
- View information about the user event scripts for a record type:
  - The **Script** column lists the script's name. Click the script's name to view its Script Deployment record. See the help topic Script Deployment.
  - The **Owner** column lists the script's owner.
  - The **API Version** column lists the script's API version.
  - The **From Bundle** column lists the number of the bundle the script is included with. Click the bundle number to view information about the bundle.
  - The **Bundle Name** column lists the name of the bundle that the script is included with. Click the bundle name to view information about the bundle.
  - The **Company Name of Bundle** column lists the name of the bundle that the company uses to identify the bundle.
  - The **Status** column lists the release status of the script, either Testing or Released. See the help topic Setting Script Deployment Status.
  - The **Countries** column lists the countries where the script is executed.
  - The **Before Load Function** column lists the names of the functions that are set to execute on the `beforeLoad` user event entry point.
  - The **Before Submit Function** column lists the names of the functions that are set to execute on the `beforeSubmit` user event entry point.
  - The **After Submit Function** column lists the names of the functions that are set to execute on the `afterSubmit` user event entry point.
  - The **Options** column lists any options in the script and their setting.

Note: For information about user event script entry points, see the help topic SuiteScript 2.x User Event Script Entry Points and API. For information about Localization Context, see Localization Context.

Tip: Deploying a large number of user event scripts of the same event type can impact performance. Avoid deploying more than 10 scripts of the same event type.

Localized Client Scripts Subtab

A maximum of 10 localized/ non-localized client scripts are supported.

Use the **Client Scripts** subtab to:

- Change the script execution order of localized client scripts. For example, have the third script execute first by moving the script to the top of the list.
- Change a script's deployment status from Testing to Released.
- Deploy a script by checking the **Deployed** box.
- View information about the client scripts for a record type:
  - The **Script** column lists the script's name. Click the script's name to view its Script Deployment record. See the help topic Script Deployment.
  - The **Owner** column lists the script's owner.
  - The **API Version** column lists the script's API version.
  - The **From Bundle** column lists the number of the bundle the script is included with. Click the bundle number to view information about the bundle.
  - The **Bundle Name** column lists the name of the bundle that the script is included with. Click the bundle name to view information about the bundle.
  - The **Company Name of Bundle** column lists the name of the bundle that the company uses to identify the bundle.
  - The **Status** column lists the release status of the script, either Testing or Released. See the help topic Setting Script Deployment Status.
  - The **Countries** column lists the countries where the script is executed.
  - The **Page Init Function** column lists the names of the functions that are set to execute on the `pageInit` client script entry point.
  - The **Save Record Function** column lists the names of the functions that are set to execute on the `saveRecord` client script entry point.

    The **Field Changed Function** column lists the names of the functions that are set to execute on the `fieldChanged` client script entry point.
  - The **Validate Line Function** column lists the names of the functions that are set to execute on the `validateLine` client script entry point.

> **Note:** For information about client script entry points, see the help topic SuiteScript 2.x Client Script Entry Points and API. For information about Localization Context, see Localization Context.

SuiteScript Monitoring with the Application Performance Management (APM)

Applies to: SuiteScript 2.x | SuiteCloud Developer

The Application Performance Management (APM) SuiteApp provides the ability to view and manage the performance of your NetSuite customizations and business critical operations through a performance dashboard. APM also includes data visualizations, page time summary, and script analysis tools to help you review and improve the speed of the NetSuite user interface.

For additional information about APM, see the help topic Application Performance Management (APM).

Setting Runtime Options

Applies to: SuiteScript 2.x | SuiteCloud Developer

If you have not already created a Script Deployment record for your SuiteScript file, see the help topic SuiteScript 2.x Entry Point Script Creation and Deployment.

After you have created a deployment for your script, see the following topics for information about setting additional deployment/runtime options:

- Setting Script Execution Event Type from the UI
- Setting Script Execution Log Levels
- Executing Scripts Using a Specific Role
- Setting Available Without Login
- Setting Script Deployment Status
- Defining Script Audience
- Creating Script Parameters Overview

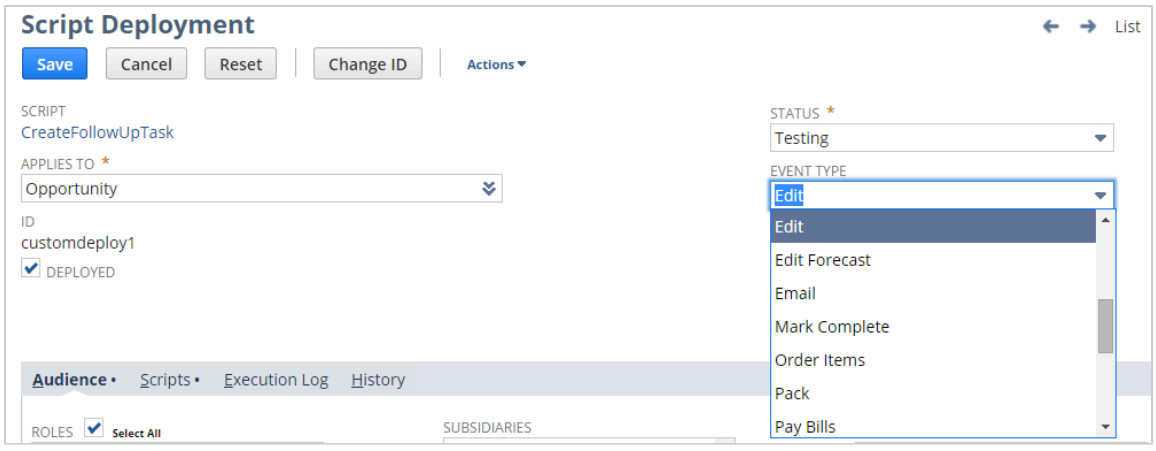
Also see these help topics for information related to script deployments, but not necessarily specific to any deployment/runtime options.

- Script Deployment
- Methods of Deploying a Script

Setting Script Execution Event Type from the UI

Applies to: SuiteScript 2.x | SuiteCloud Developer

In the Event Type field on the Script Deployment page, you can select a single event type that you want to trigger the execution of the script (see following figure). If the Event Type field is left blank, the script will execute only on the events specified in the .js script file.



Note: The Event Type field is available on Script Deployment pages for Suitelet, user event, and record-level client scripts only.

The Event Type field is useful if you want to specify a script execution context at the time of script deployment, without having to modify your .js script file. After you select an event type and click Save, the deployed script will execute only on that event, regardless of the event types specified in the .js script file.

Be aware that event types selected on the Script Deployment page take precedence over the event types specified in the .js script file. For example, if the **create** event type is specified in the script, selecting **Edit** from the Event Type field on the Script Deployment page will restrict the script from running on any event other than Edit.

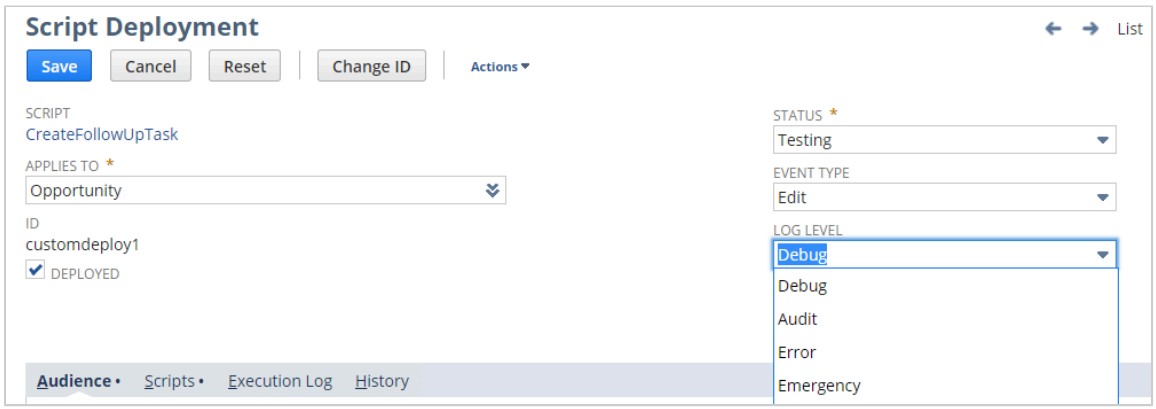
The following snippet is from a user event script. Notice that the event type specified in the code is create (`context.UserEvent.CREATE`). If the Edit event type is specified on the Script Deployment page, the script will execute only when the specified record is edited, not created.

```
function followUpCallAfterSubmit(type) {
    // execute the logic in this script onlyif a new customer is created
    if (context.type !== context.UserEventType.CREATE)
        return;
    // obtain a handle to the newly created customer record
    var customerRecord = context.newRecord;
      // remainder of script.....
   }
}
```

Setting Script Execution Log Levels

Applies to: SuiteScript 2.x | SuiteCloud Developer

In the Log Level field on the Script Deployment page, specify which log entries you want to appear on the Execution Log tab:



The Log Level field is essentially used as a basic filtering mechanism. Each log entry written with N/log Module methods specifies a log level based on the specific method used: log.audit(options), log.debug(options), log.emergency(options), log.error(options).

Set filtering by selecting one of the following log levels from the Log Level field:

- **Debug** : For scripts in testing mode. Selecting this level will show all log messages (debug, audit, error, and emergency).
- **Audit** : For scripts going into production. Selecting this level will show the events that have occurred during the processing of the script (for example, "A request was made to an external site.").
- **Error** : For scripts going into production. Selecting this level will show only unexpected script errors.
- **Emergency** : For scripts going into production. Selecting this level will show only the most critical errors in the script log.

Important: NetSuite governs the amount of logging that can be done by a company in any 60 minute time period. For complete details, see Governance on Script Logging.

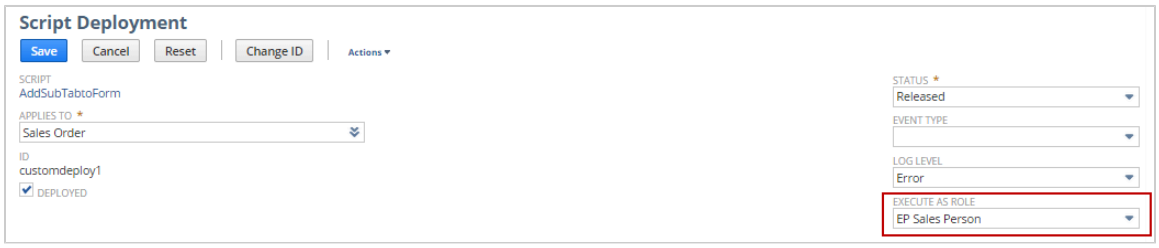
Note: The log level you specify on the Script Deployment page is independent of any error handling within your script.

See Using the Script Execution Log Tab for details on how to further customize your view of all log entries.

Executing Scripts Using a Specific Role

Applies to: SuiteScript 2.x | SuiteCloud Developer

The Execute as Role field provides role-based granularity in terms of the permissions and restrictions for executing scripts. For example, if a Sales Person role is selected in the Execute as Role field (see figure below), the script will always execute based on the permissions and restrictions assigned to the Sales Person role, even if the role of the logged in user is different. For example, if the logged in user's role is Administrator, when the script executes in this user's account, it executes based on the record-level permissions assigned to the Sales Person role and not that assigned to the administrator role.



The Current Role value in the Execute as Role field means that the script executes using the permissions of the currently logged-in user (the user whose account the script is running in).

SuiteScript developers can also select from custom roles that they have created with permissions that are specific to a script deployment.

The Execute as Role field is available on the Script Deployment pages for these script types only:

- Suitelet
- User Event
- Portlet
- Mass Update
- Workflow Action

Note: Be aware that when a script triggers other scripts, the cascading scripts will run as the role of the initial triggering script's role, not the role specified on the cascaded script's role.

For information about SuiteScript roles and permissions, see Setting Roles and Permissions for SuiteScript. For information about role restrictions in client SuiteScript, see the help topic Client Script Role Restrictions.

Testing a Script Using Different Roles

NetSuite lets you switch the deployment status to Testing when you are setting the script to execute as a different role. When you do this, the script will be triggered only by you. And, when it executes, it will execute as the role you selected.

Setting Your Script to Run With Administrative Privileges

If you want the script to execute using administrative privileges, regardless of the permissions of the currently logged-in user, select Administrator in the Execute as Role field.

There may be some scripts that are required to run with administrative privilege. For example, if you have a script that creates follow-up tasks after a sales order has been saved, and the script needs to read data from employee records, the script will not complete execution if a user's role does not have permission to access employee records. In this case, it may be appropriate to have Administrator selected in the Execute as Role field.

However, setting a script to execute as Administrator should be done with caution, as this option allows scripts to execute with privileges that the logged in user does not have. This may be appropriate for certain scripts, but there are other cases where the script performs actions that are only appropriate for certain roles.

Note: All bundle installation scripts need to execute as Administrator, so Administrator must be selected in the Execute As Role field for those scripts.

Note: Often, when scripts execute without logins (see Setting Available Without Login) or in the Web store, they tend to be implemented as an Administrator whenever any meaningful interaction with the system is required.

Setting Available Without Login

Applies to: SuiteScript 2.x | SuiteCloud Developer

You can allow Suitelet scripts to be executed without a login. Select the Available Without Login box on the Script Deployment page to allow users without an active NetSuite session to have access to the Suitelet.

Warning: Suitelets configured as available without login should not be used in integration use cases, including SDN partner SuiteApps. Suitelets configured as available without login are a violation of Built for NetSuite (BFN) standards.

To ensure that all users can access the Suitelet with or without login, check the All Roles box and clear all values from the Departments, Groups, Employees, and Partners fields under the Audience subtab on the Script Deployment page. When you select Available Without Login and save the Script Deployment record, an External URL field is displayed on the Script Deployment page. Use this URL for Suitelets you want to make available to users who do not have an active NetSuite session.

Note: The Available Without Login box is available on the Script Deployment page for Suitelets only.

The Website feature must be enabled for Clients Scripts to work in externally available Suitelets.

If you need to perform Outbound HTTPs calls in an unauthenticated client-side context, you must do so inside a Suitelet available without login and call that Suitelet using **N/https#requestSuitelet()** instead of calling one of the prohibited functions directly. See: Outbound HTTPs in an unauthenticated client-side context.

The following are some uses cases when you might want to make a Suitelet externally available:

- Hosting one-off online forms (for example, capturing partner conference registrations).
- Inbound partner communication (such as, listening for payment notification responses from PayPal or Google checkout, or for generating an unsubscribe request from email campaigns page, which requires access to account information but should not require a login or hosted website).
- For Facebook, Google, and Yahoo mashups in which the Suitelet lives in those websites but needs to communicate to NetSuite using POST requests.

Important: Be aware that the data contained within the Suitelet will be less secure when it is allowed to be accessed (using Suitelet execution) without login.

Errors Related to the Available Without Login URL

Based on the use case for your Suitelet, you will use either the internal URL or the external URL as the launching point for the Suitelet.

Some of the factors determining whether the Suitelet will deploy successfully are the dependencies between the type of URL you are referencing (internal or external), the Suitelet deployment status (Testing or Released), and whether the Select All Roles box has been selected on the Audience subtab of the Script Deployment page. The following table summarizes these dependencies.

Note: If specific roles, employees, departments, partners, or groups are selected on the Audience subtab or the All Employees or All Partners boxes are checked, external users will receive an error when accessing the Suitelet, even if the Available Without Login box is checked. To ensure access for all external users as well as internal users, check the Available Without Login box and the Select All Roles box. Leave all other fields on the Audience tab empty.

Suitelet URL Type	Deployment Status	Select All Roles box	Result
internal	Testing	not checked	Suitelet deploys successfully
internal	Testing	checked	Suitelet deploys successfully
internal	Released	not checked	Error message: You do not have privileges to view this page.

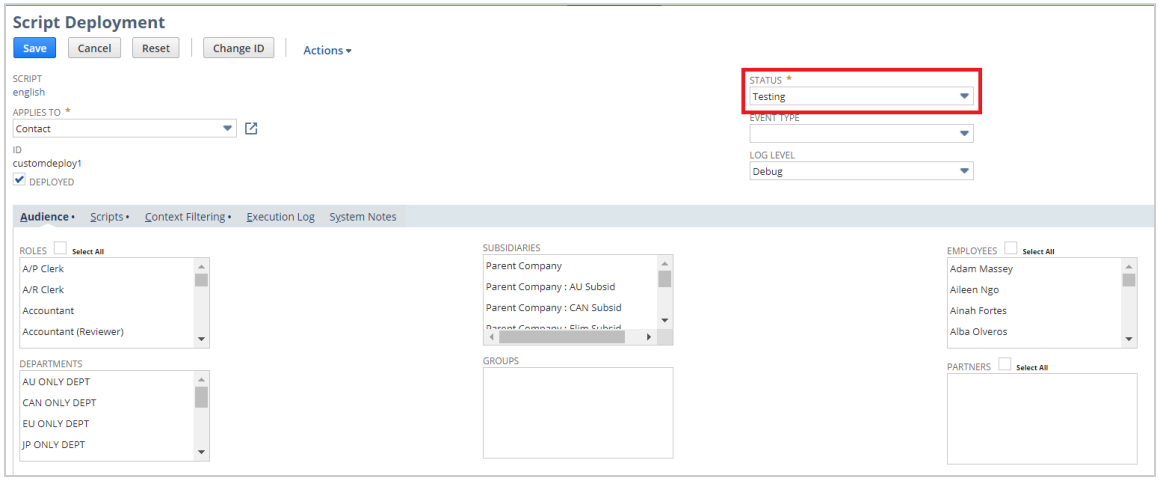
Suitelet URL Type	Deployment Status	Select All Roles box	Result
internal	Released	checked	Suitelet deploys successfully
external	Testing	not checked	Error message: You are not allowed to navigate directly to this page.
external	Testing	checked	Error message: You are not allowed to navigate directly to this page.
external	Released	not checked	Error message: You do not have privileges to view this page.
external	Released	checked	Suitelet deploys successfully

Setting Script Deployment Status

Applies to: SuiteScript 2.x | SuiteCloud Developer

A script's deployment status can be set to Testing or Released. When the status is set to Testing, the script will execute for the script owner and specified audience. Setting the status to Released means that the script will run in the accounts of all specified audience members

To set the deployment status, select either Testing or Release from the Status field on the Script Deployment page:



Note: The Testing and Released statuses do not apply to scheduled scripts. To learn about scheduled script deployment statuses, see the help topic Scheduled Script Submission.

Status Set to Testing

When a script's deployment status is set to Testing, only the script owner will be able to run the script. The script owner can test the script functionality in a different role using the Audience tab, but the script will not be available for other users until the status is set to Released. For information, see Using the Audience Subtab to Test Scripts.

Note that when using the SuiteScript Debugger to test scripts, the script's deployment status must be set to Testing. You cannot debug a Deployed script if the status has been set to Released. (See Debugging Deployed SuiteScript 1.0 and SuiteScript 2.0 Server Scripts and Debugging Deployed SuiteScript 2.1 Server Scripts for more information about using the SuiteScript Debugger to test existing scripts.)

If you are working with Suitelet Script Deployment records, also see Errors Related to the Available Without Login URL, which discusses the relevance of the Testing status as it pertains to internally and externally available Suitelets.

Note: A bundle installation script cannot execute in target accounts if its deployment status is set to Testing.

Status Set to Released

A script deployment status set to Released means that the script will run in the accounts of all specified audience members. (See Defining Script Audience for information about defining script audiences.) When the deployment status is set to Released, the script is considered to be “production ready.”

Be aware that if you do not specify any values on the Audience subtab, the script will execute only for the script owner, even if the script deployment status is set to Released.

Note: Bundle installation scripts and scheduled scripts do not have an audience. If the deployment status is set to Released for a bundle installation script, the script will execute automatically in target accounts when the associated bundle is installed or updated.

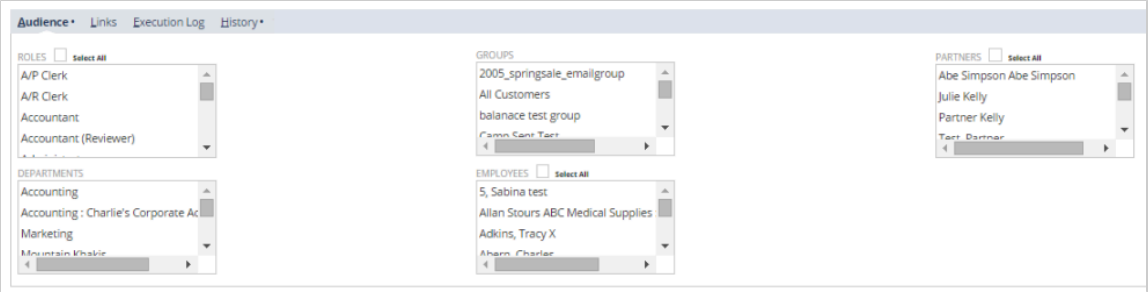
If you are working with Suitelet Script Deployment records, also see Errors Related to the Available Without Login URL, which discusses the relevance of the Released status as it pertains to internally and externally available Suitelets.

Defining Script Audience

Applies to: SuiteScript 2.x | SuiteCloud Developer

On the Audience subtab on the Script Deployment page, define the audience access for the script. When the script is deployed, it will only run in the roles specified in the Audience subtab.

The Audience subtab is included on the Script Deployment page for these script types: Suitelet, portlet, user event, global client (deployed at the record-level), workflow action. You cannot specify an audience for scheduled scripts or bundle installation scripts.



General guidelines for specifying an audience:

- If you do not specify any values on the Audience subtab, the script will execute only for the script owner even if the script deployment status is set to Released. For information about the differences between the Released and Testing deployment statuses, see Setting Script Deployment Status.
- If you choose both role and department options on the Audience subtab, a user must belong to one of the selected roles AND one of the selected departments to execute the script. If you choose options for any other combinations of types (groups, employees, and partners), a user need only belong to a selected option of one type OR of another.
- If you want the script to run in the accounts of all NetSuite users, select the Select All box next to Roles. If you want the script to run in the accounts of only specific users, select the appropriate roles, departments, groups, employees, or partners.

Be sure to save the Script Deployment record after all audience members have been defined.

If you are working with Suitelet Script Deployment records, see Errors Related to the Available Without Login URL, which discusses the relevance of the Select All box as it pertains to internally and externally available Suitelets.

Note: Mass update script deployments and mass updates can both be assigned an audience. It is the users responsibility to make sure the two audiences are in sync. For information about working with the mass update script type, see the help topic SuiteScript 2.x Mass Update Script Type.

Using the Audience Subtab to Test Scripts

If the Status field on the Script Deployment record is set to Testing, you can test scripts assigned to specific audiences if you are a member of that audience type. For information about the Testing deployment status, see Setting Script Deployment Status.

For example, if you have written a script that you want to run for everyone in the Support Management role, you can log into NetSuite and then switch to the Support Management role to test the script. If the deployment status for this script is set to Testing, the script will run for only the script owner. After you have determined that the script runs as expected for the Support Management role, you can change the script's deployment status to Released. After it is set to Released, it will run in the account for all those assigned to the Support Management role.

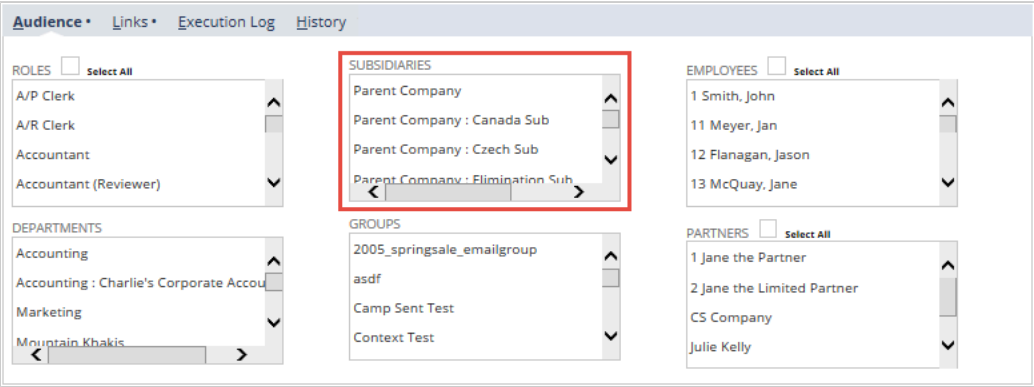
This is a good approach for script owners to verify that their script will run in the accounts for specified roles, departments, groups, employees, or partners.

Using the Audience Tab in OneWorld Accounts

Script authors/owners who are developing scripts for NetSuite OneWorld accounts can specify a script audience based on subsidiary. In OneWorld accounts, the Audience subtab includes a Subsidiaries multiselect field. After choosing subsidiaries, be sure to click Save on the Script Deployment page.

If the currently logged user is a member of a subsidiary which is selected in the multiselect field on the Script Deployment page, then the script is executed.

Script owners working in a OneWorld account that has multiple subsidiaries can select the subsidiaries they want their script to run in, and then log into an account of one of the specified subsidiaries. When a script's deployment status is set to Testing, the script will execute for the script owner and specified audience. This is a good way for script owners to verify that the script will run in accounts for specified subsidiaries.



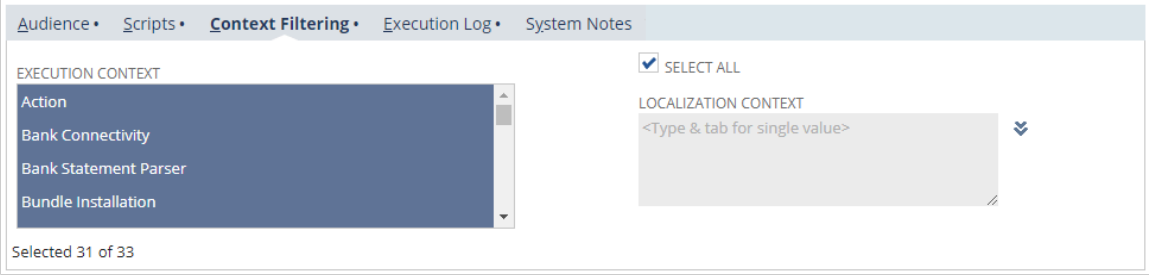
Using the Context Filtering Tab

Applies to: SuiteScript 2.x | SuiteCloud Developer

In a script deployment record, the Context Filtering tab lets you specify the contexts in which a script will run. You can use context filtering to ensure that a script runs only when necessary, which can improve performance.

You can specify the execution context and localization context for a script:

- Execution Context
- Localization Context



You can use the N/recordContext module to get the context type for a record. For more information, see the help topic N/recordContext Module.

Execution Context

Execution contexts provide information about how or when a client script or user event script is triggered to execute. For example, a script can be triggered in response to an action in the NetSuite application, or an action occurring in another context, such as a web services integration. You can use execution context filtering to ensure that your scripts are triggered only when necessary. You can specify that a script should execute only in certain contexts, and this filtering can improve performance in contexts where the script is not required. For more information, see the help topic Execution Contexts.

On the Context Filtering tab of a script deployment record, you can select the contexts in which you want your script to execute. Your script will not execute if it is triggered in a context that is not selected. By default, all contexts are selected except for Web Application and Web Store. Often, it is not required to trigger scripts in these contexts, so they are disabled by default to improve performance. If you want your script to be triggered in these contexts, be sure to select them explicitly when you create your script deployment record.

To define the execution context:

1. Define a new script deployment record or edit an existing one. For more information, see the help topic Script Deployment.
2. In the script deployment record, click the **Context Filtering** tab. By default, all execution contexts are selected except for Web Application and Web Store.
3. In the **Execution Context** field, select the contexts in which you want the script to execute. Cancel the selection of contexts in which the script should not execute.
4. Click **Save**.

Localization Context

You can define the localization context in which a client or user event script can execute. You can enable the script execution for specific localization contexts programmatically and in the UI.

Localization filtering allows you to execute a script based on the country of the active record or transaction. For a list of records that support localization context, see Records that Support Localization Context.

NetSuite automatically determines the localization context for records and transactions based on their values for country fields such as subsidiary and tax nexus. It is important to understand this determination before you set up localization context filtering for scripts. For more information, see the help topic Record Localization Context.

You can specify the execution order of localized client and user event scripts. A maximum of 10 localized/non-localized client scripts are supported. For more information, see The Scripted Records Page.

You can also search for localized scripts by using the N/query Module. For more information, see the help topic N/query Module.

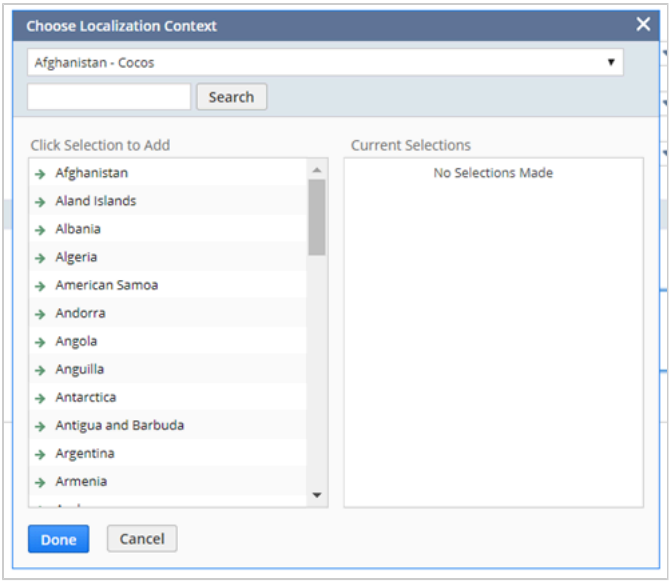
The following table shows how you can specify the localization context based on the script type.

Script Type	Defining Localization Context Filtering
SuiteScript 2.x Client Script Type	Complete the following steps to add localization context filtering to client scripts: 1. Use the localizationContextEnter(scriptContext) and localizationContextExit(scriptContext) entry points in your script. 2. Define the localization context on the Context Filtering tab on the script deployment record.
SuiteScript 2.x User Event Script Type	Define the localization context on the Context Filtering tab on the script deployment record only.

To define the localization context:

1. Define or edit your script deployment record. For more information, see the help topic Use the Script Deployment Record.
2. Click the **Context Filtering** tab. All countries are selected by default.
3. Clear the **Select All** box.
4. Click on the arrow button beside the **Localization Context** field.

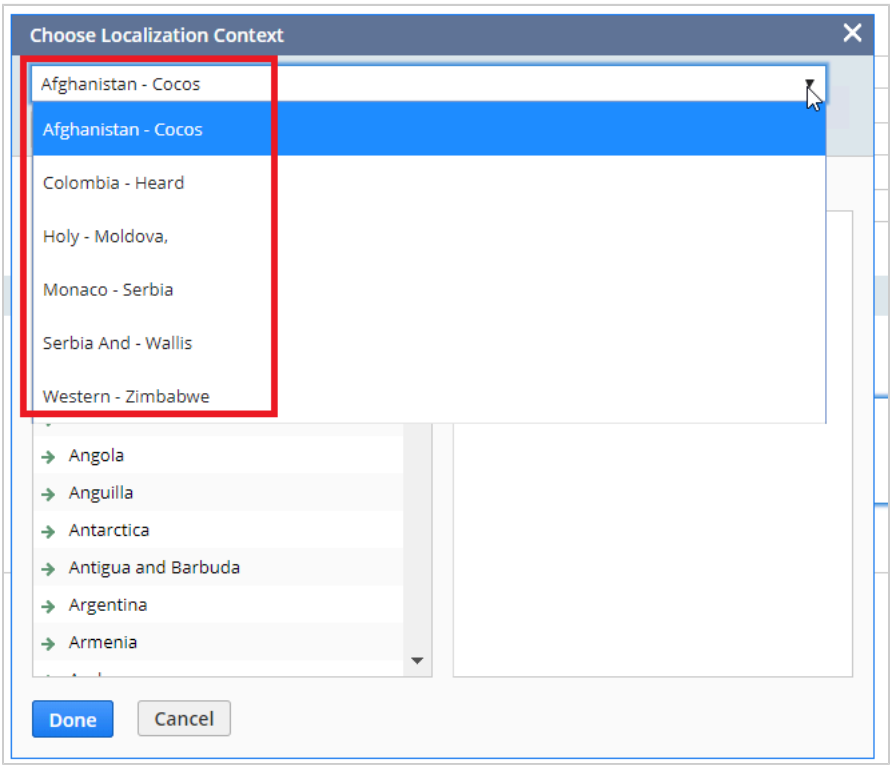
The **Choose Localization Context** popup window is displayed.



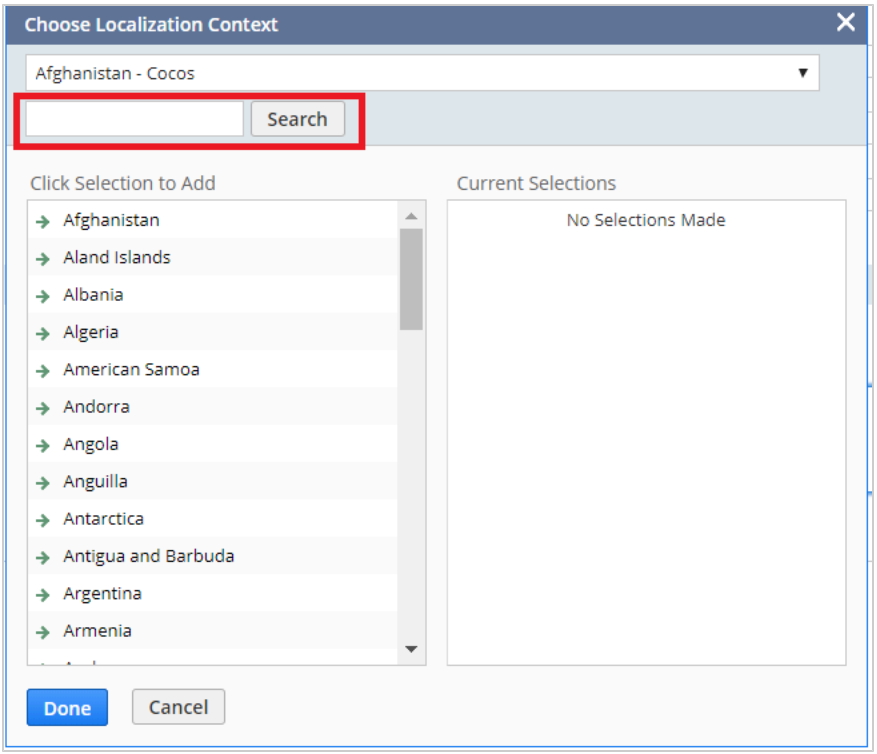
5. Select a country. The selected country is automatically displayed under Current Selections.
6. Click **Done**.

To avoid scrolling through a long list of countries, you can do any of the following:

- The drop-down list on top of the popup window displays the pagination of the countries that are listed alphabetically. Select the pagination based on the first letter of the country that you want to specify and make your selection.



- Enter the country in the **Search** field and select it.



Records that Support Localization Context

Transactions	Items	Entities	Others
Advanced Intercompany Journal Entry (advintercompanyjournalentry)	Assembly Item BOM (assemblyitembom)	Contact (contact)	Tax Type (taxtype)
Assembly Build (assemblybuild)	Description Item (descriptionitem)	Customer (customer)	Tax Code (salestaxitem)
Assembly Unbuild (assemblyunbuild)	Discount Item (discountitem)	Employee (employee)	
Bin Transfer (bintransfer)	Download Item (downloaditem)	Generic Resource (genericresource)	
Bin Worksheet (binworksheet)	Gift Certificate Item (giftcertificateitem)	Lead (lead)	
Blanket Purchase Order (blanketpurchaseorder)	Item (item)	Other Name (othername)	
Cash Refund (cashrefund)	Item Group (itemgroup)	Partner (partner)	
Cash Sale (cashsale)	Kit Item (kititem)	Project (job)	
Check (check)	Lot Numbered Build/Assembly Item (lotnumberedassemblyitem)	Project Template (projecttemplate)	
Credit Card Charge (creditcardcharge)	Lot Numbered Inventory Item (lotnumberedinventoryitem)	Prospect (prospect)	
Credit Card Refund (creditcardrefund)	Markup Item (markupitem)	Tax Group (taxgroup)	
Credit Memo (creditmemo)	Non-Inventory Part (noninventoryitem)	Vendor (vendor)	

Customer Deposit (customerdeposit)	Other Charge Item (otherchargeitem)		
Customer Payment (customerpayment)	Payment Item (paymentitem)		
Customer Payment Authorization (customerpaymentauthorization)	Sales Tax Item (salestaxitem)		
Deposit (deposit)	Serialized Build/Assembly Item (serializedassemblyitem)		
Deposit Application (depositapplication)	Serialized Inventory Item (serializedinventoryitem)		
Customer Refund (customerrefund)	Service (serviceitem)		
Estimate (estimate)	Shipping Item (shipitem)		
Expense Report (expensereport)	Subscription Plan (subscriptionplan)		
Fulfillment Request (fulfillmentrequest)	Subtotal Item (subtotalitem)		
Intercompany Journal Entry (intercompanyjournalentry)			
Intercompany Transfer Order (intercompanytransferorder)			
Inventory Adjustment (inventoryadjustment)			
Inventory Cost Revaluation (inventorycostrevaluation)			
Inventory Count (inventorycount)			
Inventory Status Change (inventorystatuschange)			
Inventory Transfer (inventorytransfer)			
Invoice (invoice)			
Item Fulfillment (itemfulfillment)			
Item Receipt (itemreceipt)			
Journal Entry (journalentry)			
Opportunity (opportunity)			
Paycheck (paycheck)			
Paycheck Journal (paycheckjournal)			
Period End Journal (periodendjournal)			
Purchase Contract (purchasecontract)			
Purchase Order (purchaseorder)			
Requisition (purchaserequisition)			
Return Authorization (returnauthorization)			
Revenue Arrangement (revenuearrangement)			
Revenue Commitment (revenuecommitment)			

Revenue Commitment Reversal (revenuecommitmentreversal)			
Sales Order (salesorder)			
Statistical Journal Entry (statisticaljournalentry)			
Store Pickup Fulfillment (storepickupfulfillment)			
Transfer Order (transferorder)			
Vendor Bill (vendorbill)			
Vendor Credit (vendorcredit)			
Vendor Payment (vendorpayment)			
Vendor Return Authorization(vendorreturnauthorization)			
Work Order (workorder)			
Work Order Close (workorderclose)			
Work Order Completion (workordercompletion)			
Work Order Issue (workorderissue)			

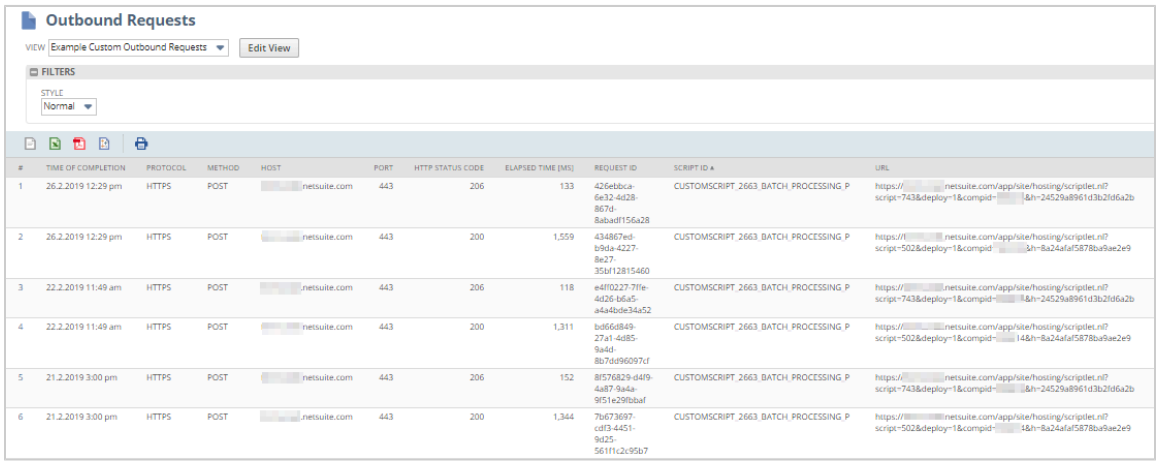
Reviewing Outbound HTTPS and SFTP Requests

Applies to: SuiteScript 2.x | SuiteCloud Developer

A log of outbound requests from NetSuite is available at Setup > Company > Communication > Outbound Request List > Search. The Outbound Requests log includes all outgoing HTTPS and SFTP requests made from your NetSuite account. Each SFTP request from SuiteScript is listed with a link to the specific script deployment that issued the request. The ability to view this type of log is important for auditing account activity. A review of this log can help you to identify requests that result in errors and requests that are not as efficient as they could be.

The Outbound Requests log displays data in a default format that includes the URL, the method, the result, and other useful details. You can modify this format by clicking Customize View. See the help topic Customizing List Views for more information.

The following figure shows a customized view in which the order of columns has been modified:



By default, the Outbound Request List search is accessible only to users with the administrator role. Administrators can provide access to additional users by assigning the Outbound Request permission, a List type permission, to other roles.

Users with the appropriate permissions also can run adhoc searches and create their own saved searches that return data on outbound requests. These capabilities are available at Setup > Company > Communication > Outbound Request List > Search.

Outbound Requests searches are also available from these menu options:

- Lists > Search > Saved Searches > New
- Reports > New Search
- Reports > Saved Searches > All Saved Searches > New

Working with the SuiteScript Records Browser

Applies to: SuiteScript 2.x | SuiteCloud Developer

To access the SuiteScript Records Browser, use the following link:

Go to the SuiteScript Records Browser

Important: When writing SuiteScript, you must use the IDs listed in the NetSuite Help Center. Although you can access the IDs by viewing page source on a NetSuite record, there is no guarantee that all IDs in the source code are supported in SuiteScript. If you create a script that references an unsupported or undocumented ID, and the ID is later changed by NetSuite, your script may break.

Not every field that appears in the SuiteScript Records Browser can be set using SuiteScript. Some fields are read only. You must check the NetSuite UI to know whether a field can be set. In general, if you can set a field in the UI, you can set it using SuiteScript. If you cannot set a field in the UI, you cannot set it using SuiteScript. However, you can still get the field's value using SuiteScript.

The SuiteScript Records Browser provides a summary of all records, fields, sublists, search joins, search filters, search columns, and record transformations that are supported in SuiteScript. Information about elements is displayed as a series of tables.

To find SuiteScript-supported records and IDs:

1. Open the SuiteScript Records Browser.

   Only records that officially support SuiteScript are listed in the SuiteScript Records Browser.
2. Click the record you want to reference in SuiteScript.

   You will see all IDs currently supported for the record.

The following table provides examples of objects and methods that use each type of ID:

ID Type	Object Examples	Method Examples
Field IDs	▪ serverWidget.Field ▪ record.Field ▪ record.Record ▪ currentRecord.CurrentRecord	▪ Record.getField(options) ▪ CurrentRecord.getField(options) ▪ Record.setValue(options) ▪ CurrentRecord.setValue(options) ▪ record.submitFields(options) ▪ search.lookupFields(options)
Sublist and sublist field IDs	▪ serverWidget.Sublist	▪ Record.commitLine(options) ▪ CurrentRecord.commitLine(options) ▪ Record.getCurrentSublistValue(options) ▪ CurrentRecord.getCurrentSublistValue(options) ▪ Record.insertLine(options) ▪ CurrentRecord.insertLine(options) ▪ Record.setSublistValue(options)

ID Type	Object Examples	Method Examples
Search join, filter, and column IDs	■ search.Search ■ search.Filter ■ search.Column ■ search.Result ■ search.ResultSet	■ search.create(options) ■ search.load(options) ■ search.createFilter(options) ■ search.createColumn(options)
Transformation IDs	■ record.Record	■ record.transform(options)

As you use the SuiteScript Records Browser, consider the following:

- You can use the Records Browser online, or you can download it. For information about downloading the browser, see the help topic Downloading the SuiteScript Records Browser.
- For information about governance applied to individual SuiteScript APIs, and about various SuiteScript script types, see SuiteScript Governance and Limits.
- Only the Records Browser for the most recent release of NetSuite is supported. Older versions may still be accessible, but they are not supported.
- Green highlighting indicates anything new in the current release.

For more details, see the following topics:

- Finding a Record or Subrecord
- Understanding the Record Summary
- Deleted Record Search

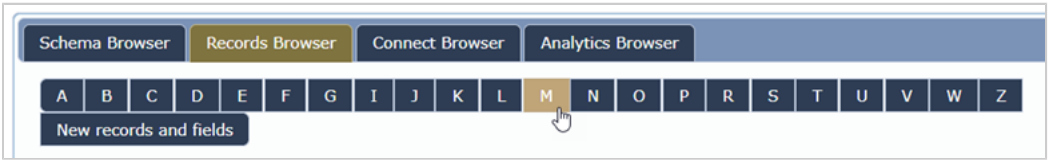
Finding a Record or Subrecord

Applies to: SuiteScript 2.x | SuiteCloud Developer

To find a record or subrecord in the SuiteScript Records Browser, use the A-Z index at the top of the browser window.

To find a record or subrecord:

1. Click the appropriate letter at the top of the browser window.

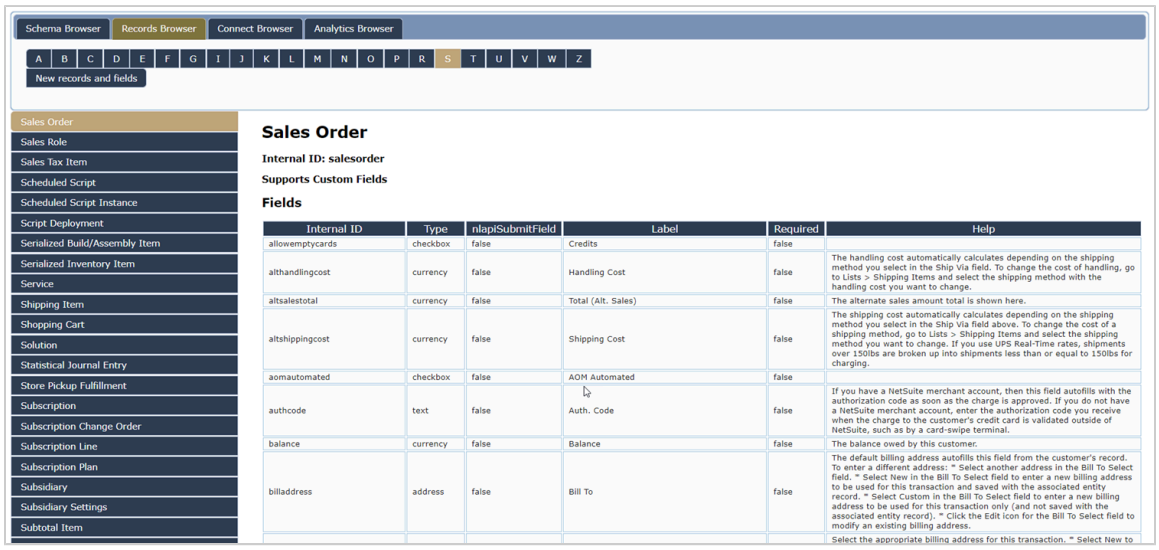


   The left pane lists all record names that begin with the letter you selected. The center pane shows the details of the first record in the list.

2. In the left pane, click the name of the record you are interested in.

   The center pane shows the details of the record.

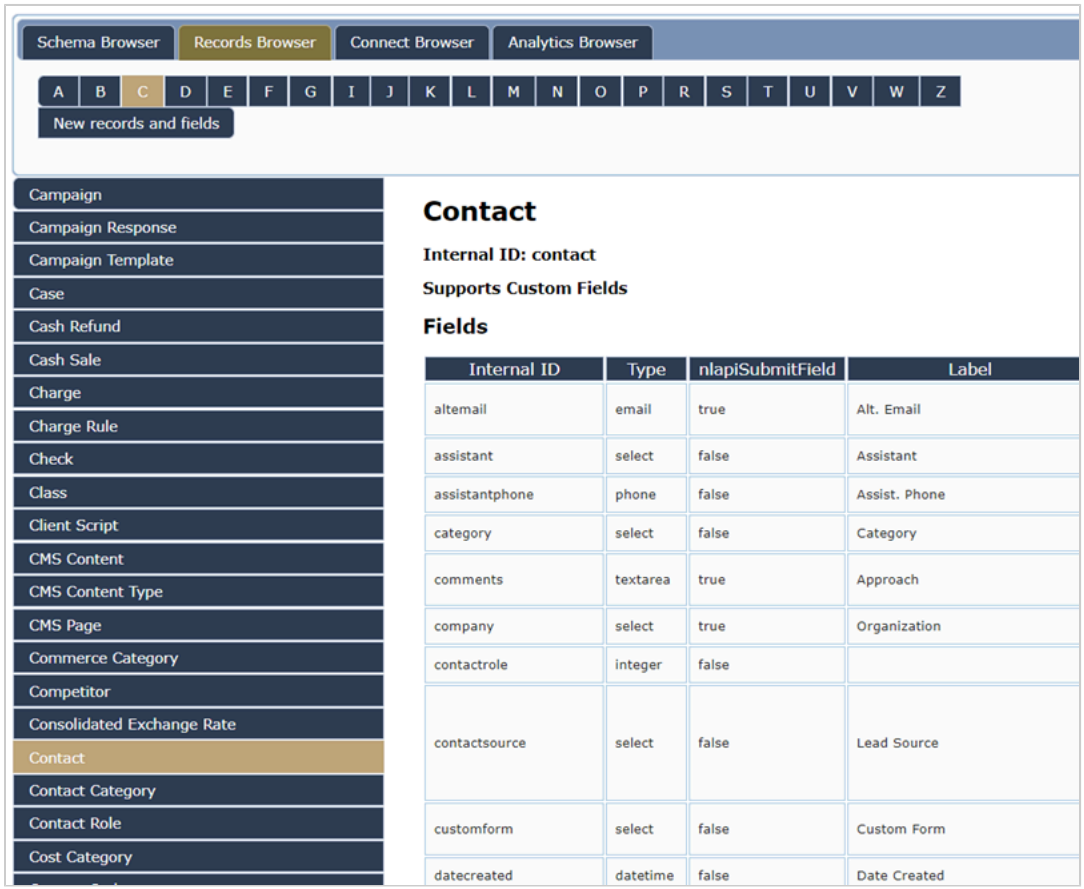
You can also get to a subrecord page from one of its parent records. Each subrecord summary field on a record page now includes a link to the subrecord page. For example, on the Sales Order page of the Records Browser, the billingaddress and shippingaddress fields include links to the address subrecord page. To go to the subrecord page, in the **Type** column, click the **summary** link.



Understanding the Record Summary

Applies to: SuiteScript 2.x | SuiteCloud Developer

The SuiteScript Records Browser includes a record summary for each record exposed to SuiteScript. For example, the following screenshot shows the record summary for a customer record:



Summary of the Record

Browser pages include the following designations for applicable records:

- **Search Only**: Indicates that the record does not support SuiteScript actions other than search actions.
- **Supports Deleted Record Search**: Indicates that a record can be used as a filter for a deleted record search. For more information, see the help topic Deleted Record Search.

For each record, the browser displays a series of tables that include the following information:

- **Fields**: The record's fields
- **Sublists**: A table representing each supported sublist
- **Tabs**: A list of tabs available on the record in the NetSuite UI
- **Search Joins**: A list of other searches you can access when searching this record
- **Search Filters**: Fields in the record that you can use as search criteria
- **Search Columns**: Fields that you can include in search results
- **Transform Types**: A list of records that this record can be transformed into using record.transform(options)

The following table describes some of the column names that these tables use:

Column Label	Description
Internal ID	The internal ID of the field.
Type	The data type of the field.
nlapiSubmitField	A boolean value indicating whether the field supports the record.submitFields(options) method, which lets you perform inline editing. The name of this column is based on the SuiteScript 1.0 API for inline editing (nlapiSubmitField). For more information, see the help topic Using Inline Editing.
Label	The label for the field as shown in the user interface.
Help	Additional details about working with the field.

Comparing SuiteScript, SOAP Web Services, and SuiteAnalytics Connect Exposure

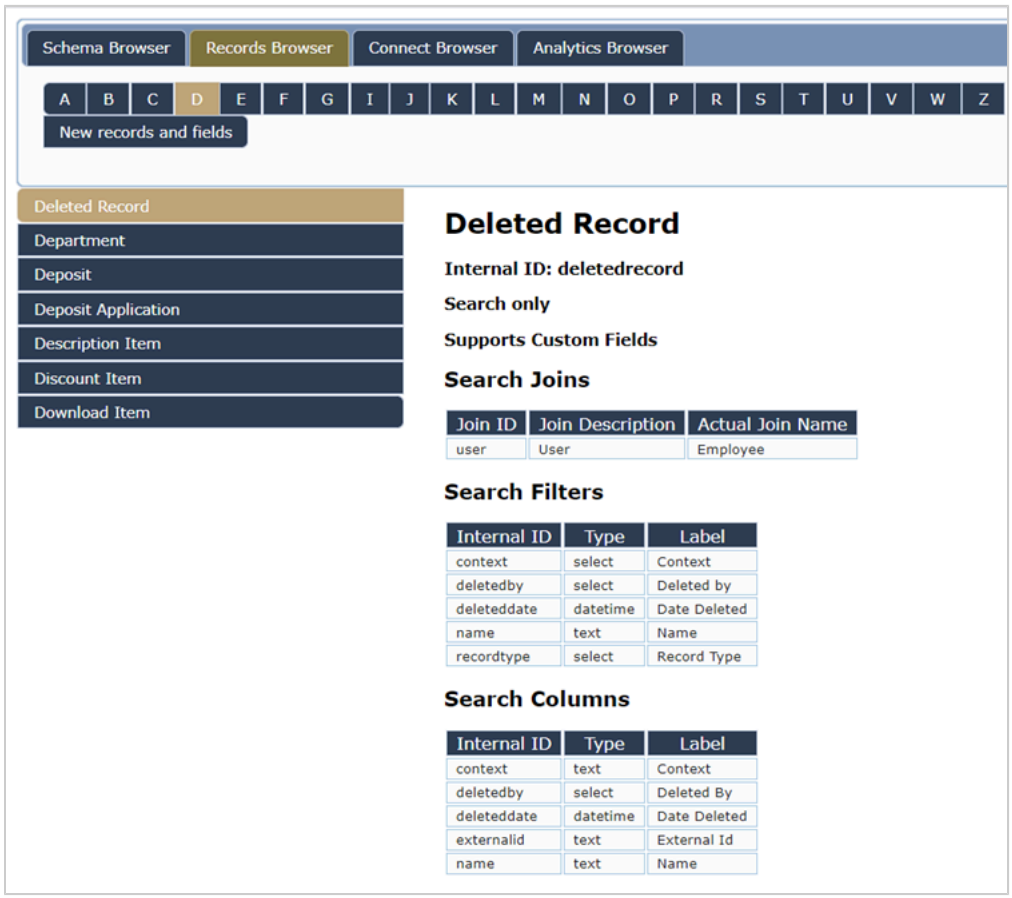
To check whether the record you are currently viewing is also supported in SOAP web services or SuiteAnalytics Connect, click the SOAP Schema Browser tab or the Connect Browser tab at the top of the page.

If the record is supported in SOAP web services or SuiteAnalytics Connect, you are directed to the corresponding page in the SOAP Schema Browser or SuiteAnalytics Connect Browser. Otherwise, you are directed to the first page of the respective browser. To compare record types support across SuiteScript, SOAP web services, and SuiteAnalytics Connect, see the help topic SuiteCloud Supported Records.

Deleted Record Search

Applies to: SuiteScript 2.x | SuiteCloud Developer

The Records Browser includes a Deleted Record page, listed under **D**. This page lists the columns and filters available for deleted record searches in SuiteScript.



You can determine whether a record can be used as a filter for deleted record searches by looking at its page in the Records Browser. The **Supports Deleted Record Search** designation appears on pages for records that can be used as filters. For information about running deleted record searches in the UI, see the help topic Searching for Deleted Records.

The retrieval of deleted records in SOAP web services is a bit different. You can use the getDeleted operation instead of running a search. For a complete list of supported record types for this operation, refer to the `DeletedRecordType` enumeration in the coreTypes spreadsheet.

Creating Script Parameters Overview

Applies to: SuiteScript 2.x | SuiteCloud Developer

In the context of SuiteScript, script parameters are similar to custom fields; they are not considered to be parameters that are passed between JavaScript functions. A script parameter can have any of the characteristics of a custom field created through point-and-click customization. Script parameters are configurable by administrators and the users of your Suite App, and are accessible programmatically through SuiteScript. Script parameters are defined on the Parameters tab of the Script record page.

Warning: Do not include confidential information in script parameters. Information saved in script parameters can be indexed by search engines and therefore be viewable by the public. This means, for example, that the information could be found in Google searches.

You should create script parameters in the following situations:

- You want part of your script to be configurable, either through script deployment or by the users of your SuiteApp. You do not need to create script parameters if your script is not designed to be configurable.
- You need to parameterize a script that was deployed multiple times. This approach makes it more convenient to customize the behavior of the script for each deployment.
- You want to configure a scheduled script. You can configure scheduled scripts by specifying the configuration parameters as arguments to task.create(options).

The advantages of using script parameters include:

- Deployment-specific parameters let you configure script behavior without having to write code. These parameters are useful when administrators deploy scripts that were installed as part of a bundle. The parameters allow administrators to control or modify the script without knowing anything about the code. Deployment-specific parameters are similar to property or configuration files that some applications use to modify behavior at runtime.
- Script parameters let you modify script behavior for troubleshooting purposes without having to change code, which is often expensive and not feasible (for example, if the original script author is unavailable).
- Script parameters give you the flexibility to handle a wide range of inputs depending on the context. For example, consider a situation in which one script is deployed to 50 different records, but requires slightly different behavior for each record. You could hard-code and deploy 50 different scripts, but they may be difficult to maintain because the code is not configurable, code might be duplicated unnecessarily, and changes in business requirements likely require code changes.

For more information, see the following help topics:

- Creating Script Parameters
- Referencing Script Parameters
- Setting Script Parameter Preferences
- Creating a Custom Field

Creating Script Parameters

Note: This topic applies to all versions of SuiteScript.

Use the following steps to create script parameters. If you are unsure how to create a script record, see the help topic Creating a Script Record.

To create a script parameter:

1. Go to Customization > Scripting > Scripts.
2. Beside the script you want to add a parameter to, click **Edit**.
3. Click the **Parameters** tab, and click **New Parameter**.
4. In the **Label** field, type the name of the parameter (custom field) as it will appear in the UI after the script is deployed.
5. In the **ID** field, type a custom ID for the script parameter.

   Script parameter IDs must be in lowercase and contain no spaces. They also cannot exceed 30 characters.

   You can leave the ID field blank to use a system-generated ID. However, it is a best practice to create your own custom ID for script parameters. Doing so will help avoid naming conflicts if you later decide to package your script.
6. In the **Type** field, select the type of the script parameter (for example, Hyperlink, Date, Free-Form Text, or Check Box).

   For more information about field types, see the help topic Field Type Descriptions for Custom Fields.
7. (Optional) If the parameter type is List/Record, in the **List/Record** field, specify the list or record.

   If you define a saved search as a List/Record script parameter, only saved searches that are public will appear in the List/Record field. For more information about working with searches using SuiteScript, see the help topic N/search Module.
8. In the **Preference** field, select the set of preferences that you want to use for the parameter.

   Depending on the value of this field, the parameter's default value is based on the values set on either the General Preferences page (for a field value of Company), Set Preferences page (for a field value of User), or the portlet setup page (for a field value of Portlet). If you do not specify a preference, the parameter is considered to be a deployment script parameter, and its value is defined on the script deployment record.

   For more information, see Setting Script Parameter Preferences.
9. (Optional) Use the **Display**, **Validation & Defaulting**, **Sourcing & Filtering**, **Access**, and **Translation** tabs to define additional values for the parameter.

   For information about defining these values, see the following help topics:

   - Setting Display Options for Custom Fields
   - Setting Validation and Defaulting Properties
   - Setting Sourcing Criteria
   - Setting Filtering Criteria
   - Restricting Access to Custom Fields
   - Adding Translations for Custom Fields
10. Click **Save** to save the parameter.

11. On the script record, click **Save** to save the script record.

Referencing Script Parameters

Applies to: SuiteCloud Developer | SuiteScript 2.x

You can use the Script.getParameter(options) method to reference script parameters that you create. This method is included in the N/runtime module, and you can use other methods in this module to work with scripts and script objects. For example, use runtime.getCurrentScript() to obtain a runtime.Script object that represents your script. For more information, see the help topic N/runtime Module.

Before you can reference script parameters in your script, you must create them using the NetSuite UI. To learn how, see Creating Script Parameters. When you create a script parameter, make sure to remember the ID that you used (or the ID that was generated automatically for you). You must use this ID in your script to obtain the value of the script parameter.

The following example from a Suitelet obtains the value of a script parameter called `custscript_mycheckbox`:

```
// Add a script parameter called Check Box Required
var myField = Form.addField({
    id: 'custscript_mycheckbox',
    label: 'Check Box Required',
    type: serverWidget.FieldType.CHECKBOX
});

// Obtain an object that represents the current script
var myScript = runtime.getCurrentScript();

// Obtain the value of the Check Box Required script parameter
var scriptParameterValue = myScript.getParameter({
    name: 'custscript_mycheckbox'
});
```

You cannot write to a script parameter using SuiteScript. Although you can read from these fields, you cannot write to them. The only time you can pass a value to a script parameter outside of the UI is when you call task.create(options) to schedule a script.

For a complete example working with script parameters in a SuiteCloud project, see the help topic Disable Tax Fields.

Setting Script Parameter Preferences

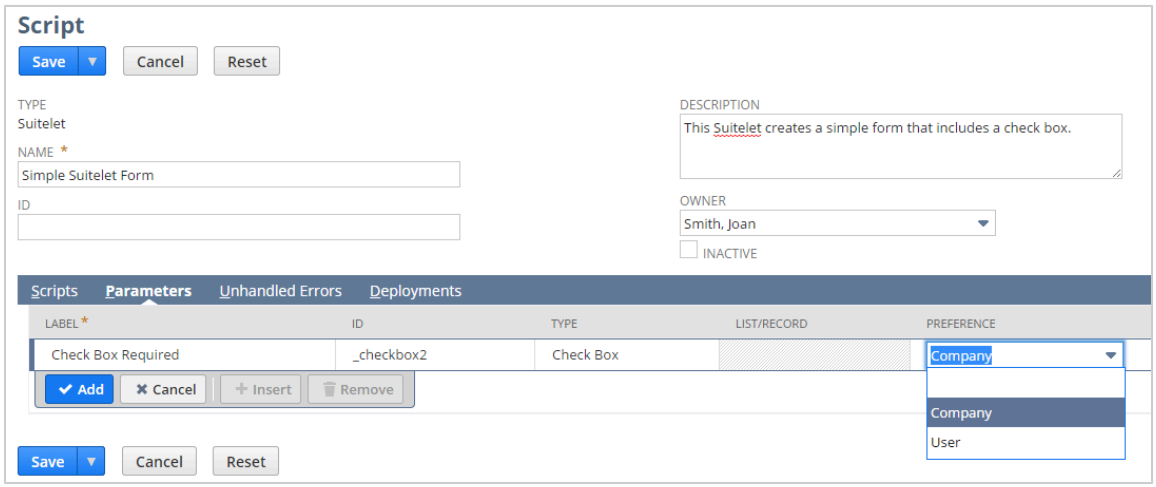
Applies to: SuiteScript 2.x | SuiteCloud Developer

As a script author, NetSuite gives you the ability to specify the preference type for each script parameter (see figure). Available preference types are:

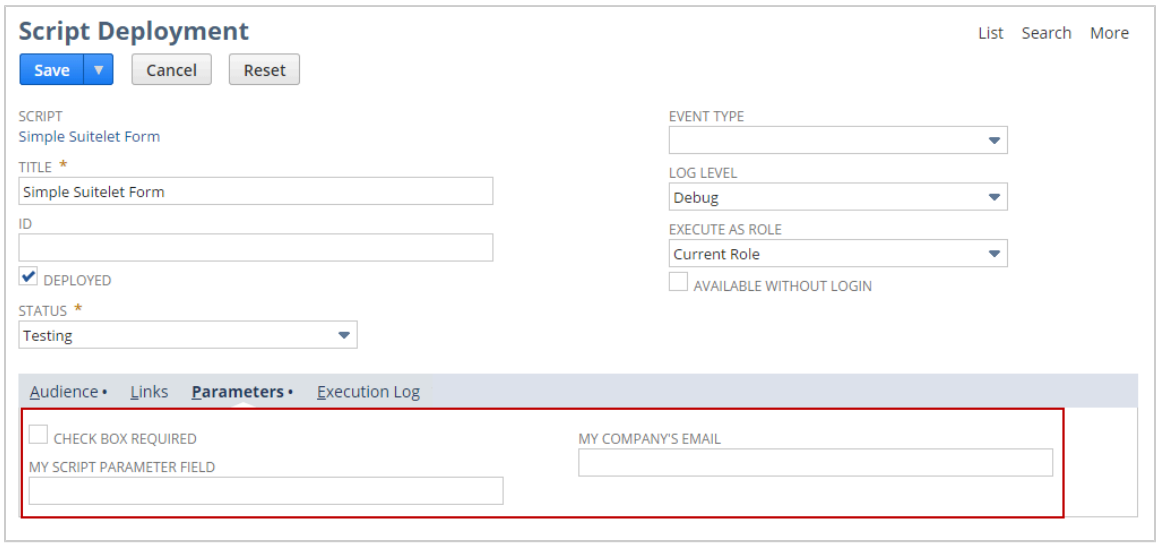
- **Company:** If the parameter preference is set to Company, the script parameter's value is read from the value specified in Setup > Company > General Preferences in the Custom Preferences tab. See the Example later in this section.
- **User:** If the preference is set to User, the parameter's value is read from the value set in Home > Set Preference in the Custom Preferences tab.

  Here, end users can override the default (company) script behavior and insert their own default value. End users do not have to manipulate a script or its deployments to change or customize the parameter.
- <blank>: If you do not set a preference, the script parameter is considered a “deployment” script parameter by default. In this case, you will define the value of the script parameter on the Parameters tab of the Script Deployment record .

Note: See Creating Script Parameters for steps on creating a script parameter. Also see Referencing Script Parameters for information about accessing script parameter values.



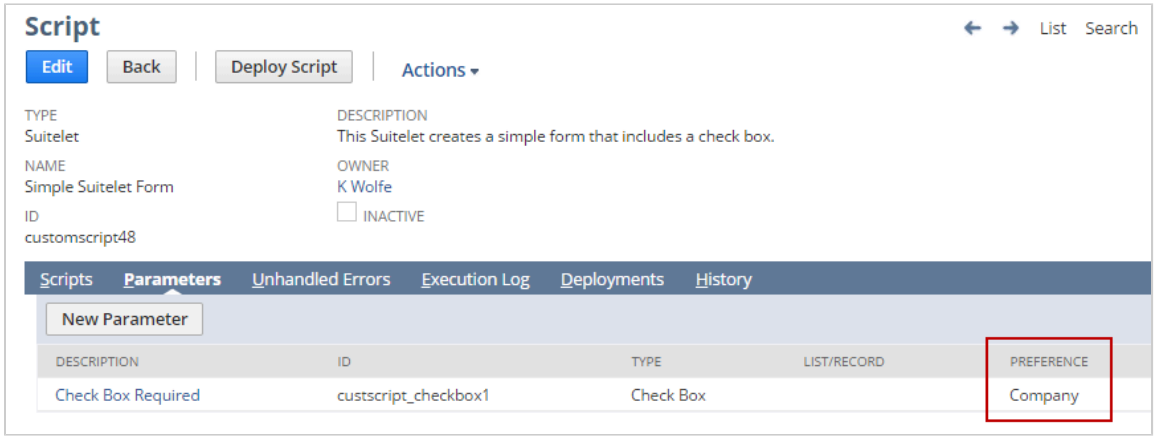
At the time these three script parameters were created, no preferences were set. In this case, parameter values are defined on the Parameters tab of the Script Deployment record.



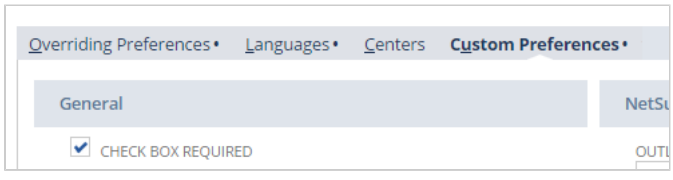
Note that users who install a bundled script that uses preferences can override the default behavior of the script and customize the script to their specific business needs. Setting preferences eliminates having to manipulate the script code or the script deployment. (For information about bundling scripts, see the *SuiteBundler Overview* topic in the NetSuite Help Center.)

Example

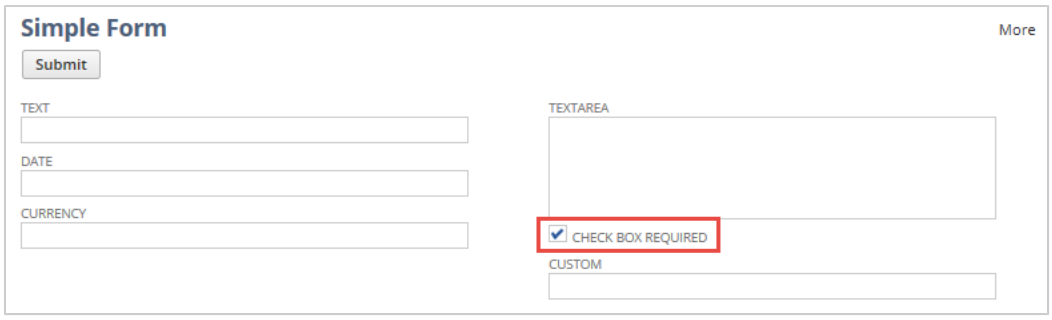
In this example, the parameter called **Check Box Required** (with the internal ID *custscript_checkboxtest2*) is set to the Company preference.



By going to Setup > Company > General Preferences in the Custom Preferences tab (see below), administrators can set the default value of this parameter for the entire company. In this example the value of the **Check Box Required** script parameter is set to T (the box is checked).



When the Suitelet that contains this box is deployed, the **Check Box Required** script parameter will appear checked.



If the **Check Box Required** parameter had been set to F (the box contained no check mark), the box would have appeared empty on the form when the Suitelet was deployed.

Script Parameter Preferences and Bundles

Bundled script parameters that have a user or company preference set are not updated in target accounts when the bundle is updated. However, script parameters that do not have a preference specified are considered part of the script deployment, and whether they are updated in target accounts when the bundle is updated depends on the setting of the related bundle object preference:

- If the preference for the bundled script is set to Update Deployments, script deployment parameters are updated in target accounts to match those in the source account.
- If the preference for the bundled script is set to Do Not Update Deployments, script deployment parameters are not updated in target accounts.

If bundle authors expect target account users to want to change parameter values for a bundled script, on the script record they should set the Preference for these parameters to be either Company or User. Target account users can then change parameter values as needed, and these values are not affected on bundle update, even if the related bundle object preference is set to Update Deployments.

To prevent changes to target account script deployment parameters that do not have a preference set, set the related bundle object preference to Do Not Update Deployments.

For more information about bundle object preferences, see the help topic Bundle Object Preferences.

SuiteScript IDs

Applies to: SuiteScript 2.x | SuiteCloud Developer

IDs are used for several different things including permissions, features, names, tasks, and buttons.

- Permission Names and IDs
- Feature Names and IDs
- Preference Names and IDs
- Task IDs
- Button IDs

Permission Names and IDs

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following table provides permission names and IDs associated with each NetSuite feature. You can use the permission ID to return the permission levels that have been specified in your account by calling User.getPermission(options) in the N/runtime Module.

The following link provides access to a Microsoft Excel worksheet listing the usage of most NetSuite permissions: NetSuitePermissionsUsage.xls. You can use this list to understand the implications of assigning a specific permission, or to find the permission required to provide access to a specific task or page. For more information, see the help topic Permissions Documentation.

Permission ID	Permission Name	Feature	Valid Levels
ADMI_ACCOUNTING	Accounting Management	Accounting	None, Full
ADMI_ACCOUNTINGBOOK	Accounting Book	Accounting	None, View, Create, Edit, Full
ADMI_ACCOUNTINGLIST	Accounting Lists	Accounting	None, View, Create, Edit, Full
ADMI_ACCTPERIODS	Manage Accounting Periods	Accounting Periods	None, View, Full
ADMI_ACCTSETUP	Set Up Accounting	Accounting	None, Full
ADMI_ACCTSETUP	Accounting Preferences	Accounting	None, Full
ADMI_ADMINDOCSEU	Admindocs EU	—	None, Full
ADMI_ADMINDOCSNA	Admindocs NA	—	None, Full
ADMI_ADMINDOCSOTHER	Admindocs OTHER	—	None, Full
ADMI_ACH	Set Up ACH Processing	ACH Processing	None, Full
ADMI_ADVANCED_ORDER_MANAGEMENT	Advanced Order Management	Advanced Order Management	None, Full
ADMI_ADVANCED_TEMPLATES	Advanced PDF/HTML Templates	Advanced Printing	None, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_ALLOW_JS_HTML_UPLOAD	Allow JS / HTML Uploads	Documents	None, Full
ADMI_ALLOWNONGLCHANGES	Allow Non G/L Changes	Accounting Periods	None, Full
ADMI_ANALYTICS	Analytics Administrator	—	None, Full
ADMI_APP_DEPLOYMENT	SuiteApp Deployment	SuiteApp	None, Full
ADMI_APPDEFPKG	App Definitions and Packages	—	None, Full
ADMI_APPPUBLISHER	Application Publishers	SSP Applications	None, Full
ADMI_AUDITLOGIN	View Login Audit Trail	Login Audit Trail	None, Full
ADMI_BACKUPEXPORT	Backup Your Data	CSV Export	None, Full
ADMI_BALANCE_TRX_BY_SEGMENTS	Balance Transactions by Segments	Balancing Segments	None, View, Create
ADMI_BANK_CONNECTIVITY_CONFIG	Bank Connectivity Plug-In Configuration	—	None, Full
ADMI_BILLINGINFO	Billing Information	Allow Multiple Users	None, Full
ADMI_BLCGA	Balance Location Costing Group Accounts	Group Average Costing	None, Full
ADMI_BUNDLER	SuiteBundler	SuiteBundler	None, Full
ADMI_BUNDLER	SuiteApp Marketplace	SuiteBundler	None, Full
ADMI_BUNDLERAUDITTRAIL	SuiteBundler Audit Trail	SuiteBundler	None, Full
ADMI_BUNDLERMANUP	SuiteBundler Upgrade Install Base	SuiteBundler	None, Full
ADMI_CAMPAIGNEMAIL	Set Up Campaign Email Addresses	Marketing Automation	None, Full
ADMI_CAMPAIGNSETUP	Setup Campaigns	Marketing Automation	None, Full
ADMI_CASEALERT	Case Alerts	Customer Support and Service	None, View, Full
ADMI_CASEFORM	Online Case Form	Customer Support and Service	None, Full
ADMI_CASEISSUE	Support Case Issue	Customer Support and Service	None, Full
ADMI_CASEORIGIN	Support Case Origin	Customer Support and Service	None, Full
ADMI_CASEPRIORITY	Support Case Priority	Customer Support and Service	None, Full
ADMI_CASERULE	Support Case Territory Rule	Customer Support and Service	None, View, Create, Edit, Full
ADMI_CASESTATUS	Support Case Status	Customer Support and Service	None, Full
ADMI_CASETERRITORY	Support Case Territory	Customer Support and Service	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_CASETYPE	Support Case Type	Customer Support and Service	None, Full
ADMI_CENTERLINK	Custom Center Link	Custom Centers	None, Full
ADMI_CERTIFICATES	Certificate Management	—	None, View, Create, Edit, Full
ADMI_CLASSESTOLOCS	Convert Classes to Locations	Locations	None, Full
ADMI_CLASSSEGMENTMAPPING	Class Segment Mapping	Classes	None, Full
ADMI_CLASSSEGMENTMAPPING	Class Mapping	Multi Book Version 2	—
ADMI_CLOSEPERIOD	Lock Transactions	Accounting Periods	None, Full
ADMI_COMMERCECATEGORY	Commerce Categories	—	None, Full
ADMI_COMMISSIONSETUP	Commission Feature Setup	Commissions	None, Full
ADMI_COMPANY	Company Information	Company Setup	None, Full
ADMI_CONVERTCLASSES	Convert Classes to Departments	Departments	None, Full
ADMI_CONVERTLEAD	Lead Conversion Mapping	Sales Force Automation	None, Full
ADMI_COPYPROJECTTASK	Copy Project Tasks	Project Management	None, Full
ADMI_CREATEJOBSFROMSALESTRANS	Create Jobs from Sales Transactions	Project Management	None, Full
ADMI_CREDITCARD	Credit Card Processing	Credit Card Payments	None, Full
ADMI_CRMLIST	CRM Lists	Opportunities	None, View, Create, Edit, Full
ADMI_CROSSCHARGE	Manage Cross Charge Automation	Intercompany Framework	View, None, Full
ADMI_CSVIMPORTPREF	Set Up CSV Preferences	Accounting	None, Full
ADMI_CUSTADDRESSFORM	Custom Address Form	Custom Forms	None, View, Create, Edit, Full
ADMI_CUSTBODYFIELD	Custom Body Fields	Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTCATEGORY	Custom Center Categories	Custom Centers	None, View, Create, Edit, Full
ADMI_CUSTCENTER	Custom Centers	Custom Centers	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_CUSTCOLUMNFIELD	Custom Column Fields	Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTEMAILLAYOUT	Custom HTML Layouts	Custom PDF/HTML Templates	None, View, Create, Edit, Full
ADMI_CUSTENTITYFIELD	Custom Entity Fields	Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTENTRYFORM	Custom Entry Forms	Custom Forms	None, View, Create, Edit, Full
ADMI_CUSTEVENTFIELD	Custom Event Fields	Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTFIELD	Custom Fields	SuiteFlow Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTFIELDTAB	Custom Subtabs	Custom Subtabs	None, View, Create, Edit, Full
ADMI_CUSTFORM	Custom Transaction Forms	Custom Forms	None, View, Create, Edit, Full
ADMI_CUSTITEMFIELD	Custom Item Fields	Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTITEMNUMBERFIELD	Custom Item Number Fields	Inventory	None, View, Create, Edit, Full
ADMI_CUSTLAYOUT	Custom PDF Layouts	Custom PDF/HTML Templates	None, View, Create, Edit, Full
ADMI_CUSTLIST	Custom Lists	Custom Lists	None, View, Create, Edit, Full
ADMI_CUSTOMERFORM	Online Customer Form	Sales Force Automation	None, View, Create, Edit, Full
ADMI_CUSTOMERRULE	Sales Territory Rule	Sales Force Automation	None, View, Create, Edit, Full
ADMI_CUSTOMER_SEGMENTS	Customer Segments Manager	—	None, Full
ADMI_CUSTOMIZEDFIELDLEVELHELP	Customize Field Level Help	SuiteBuilder Customized Field Level Help	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_CUSTOMSCRIPT	SuiteScript	Client SuiteScript	None, View, Create, Edit, Full
ADMI_CUSTOMSUBLIST	Custom Sublists	Custom Sublists	None, View, Create, Edit, Full
ADMI_CUSTOTHERFIELD	Other Custom Fields	Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTRECORD	Custom Record Types	Custom Records	None, View, Create, Edit, Full
ADMI_CUSTRECORDFORM	Online Custom Record Form	Custom Forms	None, View, Create, Edit, Full
ADMI_CUSTSECTION	Custom Center Tabs	Custom Records	None, View, Create, Edit, Full
ADMI_CUSTTASKS	Custom Center Links	Custom Records	None, View, Create, Edit, Full
ADMI_CUSTTRANFIELD	Custom Transaction Fields	Custom Fields	None, View, Create, Edit, Full
ADMI_CUSTTRANSACTION	Custom Transaction Types	Custom Transactions	None, View, Create, Edit, Full
ADMI_DELETEDRECORD	Deleted Records	Search for Deleted Records	None, Full
ADMI_DEPTSEGMENTMAPPING	Department Segment Mapping	Departments	None, Full
ADMI_DEPTSEGMENTMAPPING	Department Mapping	Multi Book Version 2	None, Full
ADMI_DEVICE_ID	Device ID Management	Suite Commerce InStore	None, Full
ADMI_DOMAINS	Set Up Domains	SuiteCommerce	None, Full
ADMI_DUPLICATESETUP	Duplicate Detection Setup	Duplicate Detection & Merge	None, Full
ADMI_EMPLCATEGORY	Publish Employee List	SuiteCommerce	None, Full
ADMI_EMPLOYEE_EXPENSE_SOURCE	Employee Expense Sources	Expense Reports	None, View, Create, Edit, Full
ADMI_EMPLOYEECENTERPUBLISHING	Employee Center Publishing	—	None, Full
ADMI_EMPLOYEELIST	Other Lists	Accounting	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_ENABLEFEATURES	Enable Features	Enable Company Features	None, Full
ADMI_ENTITYACCOUNTMAPPING	Entity Account Mapping	Accounting	None, Full
ADMI_ENTITYSTATUS	Customer Status	Customer Relationship Management	None, Full
ADMI_ESCALATIONRULE	Escalation Assignment Rule	Customer Support and Service	None, View, Create, Edit, Full
ADMI_ESCALATIONTERRITORY	Escalation Assignment	Customer Support and Service	None, View, Create, Edit, Full
ADMI_EXPENSEREPORTPOLICY	Expense Report Policies	Expense Reports	None, View, Full
ADMI_EXPORTIIF	Export as IIF	Accounting	None, Full
ADMI_FFTEXCEPTIONREASON	Fulfillment Exception Reason	—	None, View, Create, Edit, Full
ADMI_FINANCIALINSTITUTION	Financial Institution Records	—	None, Full
ADMI_FINCHARGEPREF	Finance Charge Preferences	A/R	None, Full
ADMI_GAINLOSSACCTMAPPING	Foreign Currency Variance Mapping	Accounting	None, Full
ADMI_GLOBALACCOUNTMAPPING	Global Account Mapping	Accounting	None, Full
ADMI_IMPORTCSVFILE	Import CSV File	CSV Import	None, Full
ADMI_IMPORTOVERRIDESSTRIG	Control SuiteScript and Workflow Triggers per CSV Import		None, Full
ADMI_IMPORTXML	Set Up ADP Payroll	Import ADP Payroll	None, Full
ADMI_INTEGRAPP	Integration Application	Integration	None, Full
ADMI_ISSUESETUP	Issue Setup	Issue Management	None, Full
ADMI_ISSUESHOWSTOPPER	Mark Issue As Showstopper	Issue Management	None, Full
ADMI_ITEMACCOUNTMAPPING	Item Account Mapping	Accounting	None, Full
ADMI_KERNEL	Core Administration Permissions	Core Administration Permissions	None, Full
ADMI_KEYS	Key management	—	None, View, Create, Edit, Full
ADMI_KNOWLEDGEBASE	Publish Knowledge Base	Knowledge Base	None, View, Create, Edit, Full
ADMI_KPIREPORT	KPI Scorecards	KPI Scorecards	None, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_LOCATIONCOSTINGGROUP	Location Costing Group	Item Record Management	None, Full
ADMI_LOCSEGMENTMAPPING	Location Segment Mapping	Custom Segments	None, Full
ADMI_LOCSEGMENTMAPPING	Location Mapping	Multi Book Version 2	None, Full
ADMI_LOGIN_OAUTH	Log in using Access Tokens	TBA	None, Full
ADMI_LOGIN_OAUTH2	Log in using OAuth 2.0 Access Tokens	OAuth 2.0	None, Full
ADMI_MANAGECUSTOMSEGMENTS	Custom Segments	Custom Segments	None, View, Create, Edit, Full
ADMI_MANAGE_OAUTH2	OAuth 2.0 Authorized Applications Management	OAuth 2.0	None, Full
ADMI_MANAGE_OAUTH_TOKENS	Access Token Management	Token-based Authentication	None, Full
ADMI_MANAGE_OWN_OAUTH_TOKENS	User Access Tokens	Token-based Authentication	None, Full
ADMI_MANAGE_RESTRICTIONS	Manage Custom Restrictions	Advanced Employee Permissions	None, View, Edit, Full, Create
ADMI_MANAGEPERMISSIONS	Manage Custom Permissions	Allow Multiple Users	None, Full
ADMI_MANAGEROLES	Bulk Manage Roles	Show Role Differences	None, Full
ADMI_MANAGEUSERS	Manage Users	Managing Users	None, Full
ADMI_MANUFACTURING	Manufacturing Preferences	Assembly Items	None, Full
ADMI_MHLEVEL	Merchandise Hierarchy Level	—	None, Full
ADMI_MHNODE	Merchandise Hierarchy Node	—	None, Full
ADMI_MHVERSION	Merchandise Hierarchy Version	—	None, Full
ADMI_MIGRATEREVARRNGANDPLAN	Migrate Revenue Arrangements and Plans	Advanced Revenue Management	None, Full
ADMI_MOBILE_ACCESS	Mobile Device Access	—	None, Full
ADMI_NSASOIDCPROVIDER	OIDC Provider Setup		None, Full
ADMI_NUMBERING	Auto-Generated Numbers		None, View, Edit, Full
ADMI_OIDC	OpenID Connect (OIDC) Single Sign-On	—	None, Full
ADMI_OIDCSETUP	Set Up OpenID Connect (OIDC) Single Sign-On	—	None, Full
ADMI_OPENIDSSO	OpenID Single Sign-on	—	None, Full
ADMI_OPENIDSSOSETUP	Set Up OpenID Single Sign-on	—	None, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_ORDERALLOCATIONSTRATEGY	Order Allocation Strategy	—	None, View, Create, Edit, Full
ADMI_ORDERPROMISING	Order Promising	—	None, View, Full
ADMI_OUTLOOKINTEGRATION	Outlook Integration 2.0	Outlook Integration	None, Full
ADMI_OUTLOOKINTEGRATION_V3	Outlook Integration 3.0	Outlook Integration	None, Full
ADMI_PARTNERCONTRIBUTION	Partner Contribution	—	None, Full
ADMI_PAYMENT_LINK_SETUP	Set Up Payment Link		None, Full
ADMI_PAYROLL	Set Up Payroll	Payroll	None, Full
ADMI_PENDINGBOOKJOURNAL	Allow Pending Book Journal Entry	—	None, Full
ADMI_PERIODCLOSING	Period Closing Management	—	None, Full
ADMI_PERIODOVERRIDE	Override Period Restrictions	—	None, Full
ADMI_PI_REMOVAL_CREATE	Remove Personal Information Create	—	None, Full
ADMI_PI_REMOVAL_RUN	Remove Personal Information Run	—	None, Full
ADMI_PROJECT_ACCOUNTING_SETUP	Project Profitability Setup	Project Management	None, Full
ADMI_PROJECT_ACCOUNTING_SETUP	Project Profitability	Advanced Project Profitability	None, Full
ADMI_PROVISION	Provisioning	NLCORP feature	None, View, Full
ADMI_REPOGROUPS	Financial Statement Sections	Accounting	None, Full
ADMI_REPOLAYOUTS	Financial Statement Layouts	Accounting	None, Full
ADMI_RESTWEBSERVICES	REST Web Services	REST Web Services	None, Full
ADMI_REVIEW_CUSTOM_GL_RUNS	Review Custom GL plug-in executions	Custom GL Lines Plug-in	None, Full
ADMI_SALESCHANNEL	Sales Channel	—	None, View, Edit, Full, Create
ADMI_SALESTERRITORY	Sales Territory	Sales Force Automation	None, View, Create, Edit, Full
ADMI_SAMLSSO	SAML Single Sign-on	SAML Single Sign-on	None, Full
ADMI_SAMLSSOSETUP	Set Up SAML Single Sign-on	SAML Single Sign-on	None, Full
ADMI_SAVEDASHBOARD	Publish Dashboards	Publishing Dashboards	None, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_SETUPCOMPANY	Set Up Company	Company Preferences	None, View, Full
ADMI_SETUPIMAGERESIZE	Set Up Image Resizing	SuiteCommerce Advanced	None, Full
ADMI_SETUPYEARSTATUS	Set Up Year Status	Accounting	None, Full
ADMI_SFASETUP	Sales Force Automation Setup	Sales Force Automation	None, Full
ADMI_SITEMANAGEMENT	Web Site Management	Web Site	None, Full
ADMI_STATETAXIMPORT	Import State Sales Tax	Accounting	None, Full
ADMI_STORESEARCH	Site Search	—	None, Full
ADMI_STORESETUP	Set Up Web Site	Web Site	None, Full
ADMI_SUBLIST	Custom Sublist	Allow Multiple Users	None, View, Create, Edit, Full
ADMI_SUBSIDIARYHIERARCHYMOD	Subsidiary Hierarchy Modification	—	None, View, Edit
ADMI_SUBSIDIARYSETTINGSMANAGER	Subsidiary Settings Manager	—	None, View, Edit
ADMI_SUITE_OAX_CONNECTOR	NSAW Connector Administrator		None, Full
ADMI_SUITEANALYTICSCONNECT	SuiteAnalytics Connect	SuiteAnalytics Connect	None, Full
ADMI_SUITEAPP_MANAGEMENT	SuiteApp Management	SuiteApp Control Center	None, Full
ADMI_SUITECOMMERCEANALYTICS	SuiteCommerce Analytics		None, View, Full
ADMI_SUITESIGNON	SuiteSignOn	Outbound Single Sign-on	None, Full
ADMI_SUITE_OAX_CONNECTOR	OAX Connector Administrator	NetSuite Analytics Warehouse	None, Full
ADMI_SUPPLYALLOCATIONSETUP	Supply Allocation Setup	—	None, View, Create, Edit, Full
ADMI_SUPPORTSETUP	Support Setup	Customer Support and Service	None, Full
ADMI_SWAPPRICES	Swap Prices Between Price Levels	Multiple Prices	None, Full
ADMI_TAXMIGRATION	SuiteTax Migration	—	None, Full
ADMI_TIMEMODIFICATION	Bulk Time Entry Modification	—	None, Full
ADMI_TAXPERIODS	Manage Tax Reporting Periods	Accounting	None, Full
ADMI_TEAMSELLINGCONTRIBUTION	Team Selling Contribution	Team Selling	None, Full
ADMI_TELEPHONY_SETUP	Telephony Integration	Telephony	None, Full

Permission ID	Permission Name	Feature	Valid Levels
ADMI_TRAN_ACCOUNTING_RULES	Transaction Accounting Rules	—	None, Full
ADMI_TRANSITEMTXT	Translation	Multi-Language	None, Full
ADMI_TRANSLATION	Manage Translation	Multi-Language	None, Full
ADMI_TSTDRV_MASTER	Testdrive Masters	NLCORP feature	None, Full
ADMI_TWOFACTORAUTH	Two-Factor Authentication	Two-Factor Authentication	None, Full
ADMI_TWOFACTORAUTHBASE	Two-Factor Authentication base	Two-Factor Authentication	None, Full
ADMI_UNCATSITEITEMS	Uncategorized Presentation Items	Web Store	None, Full
ADMI_UPDATEPRICES	Update Prices	Item Record Management	None, Full
ADMI_UPSELLSETUP	Upsell Setup	Upsell Manager	None, Full
ADMI_WEBSERVICES	SOAP Web Services	SuiteTalk	None, Full
ADMI_WEBSERVICES	Web Services	Internal Web Services	None, Full
ADMI_WEBSERVICESLOG	View SOAP Web Services Logs	SuiteTalk	None, Full
ADMI_WEBSERVICESLOG	View Web Services Logs	Internal Web Services	None, Full
ADMI_WEBSERVICESSETUP	Set Up SOAP Web Services	SuiteTalk	None, Full
ADMI_WEBSERVICESSETUP	Set Up Web Services	Internal Web Services	None, Full
ADMI_WORKFLOW	Workflow	SuiteFlow	None, Full
ADMINDOCS	Admindocs	NLCORP feature	None, Full
LIST_ACCOUNT	Accounts	Accounting	None, View, Create, Edit, Full
LIST_ACH	Automated Clearing House	Payment Instruments	None, View, Create, Edit, Full
LIST_ALLGOVERNMENTISSUEDIDS	Advanced Government-Issued IDs	Advanced Government-Issued ID Tracking	None, View, Create, Edit, Full
LIST_ALLOCSCHEDULE	Allocation Schedules	Expense Allocation	None, View, Create, Edit, Full
LIST_AMORTIZATION	Amortization Schedules	Amortization	None, View, Create, Edit, Full
LIST_BASICGOVERNMENTISSUEDIDS	Basic Government-Issued IDs	Basic Government-Issued ID Tracking	None, View, Create, Edit, Full
LIST_BIG_SEARCH	Persist Search	Search	None, Create

Permission ID	Permission Name	Feature	Valid Levels
LIST_BILLINBOUNDSHIPMENT	Bill Inbound Shipment	Inbound Shipment Management	None, View, Create, Edit, Full
LIST_BILLINGSCHEDULE	Billing Schedules	SuiteBilling	None, View, Create, Edit, Full
LIST_BILLOFDISTRIBUTION	Bill Of Distribution	Advanced Inventory Management	None, View, Create, Edit, Full
LIST_BILLOFMATERIALSINQUIRY	Bill Of Materials Inquiry	Assembly Items	None, View, Create, Edit, Full
LIST_BOM	Bill of Materials	Assembly Items	None, View, Create, Edit, Full
LIST_BILLCAPTURE	Scanned Vendor Bills	—	None, View, Create, Edit, Full
LIST_BIN	Bins	Accounting	None, View, Create, Edit, Full
LIST_BONUS	Bonus	—	View, None, Create, Full, Edit
LIST_BONUSTYPE	Bonus Types	—	View, None, Create, Full, Edit
LIST_CALENDAR	Calendar	Calendar Preferences	None, View, Create, Edit, Full
LIST_CALL	Phone Calls	Phone Calls	None, View, Create, Edit, Full
LIST_CAMPAIGN	Marketing Campaigns	Marketing Automation	None, View, Create, Edit, Full
LIST_CAMPAIGNHISTORY	Campaign History	Marketing Automation	None, View, Create, Edit, Full
LIST_CARDHOLDERAUTHENTICATION	Cardholder Authentications	Credit Card Payments	None, View, Edit, Full, Create
LIST_CARDHOLDERAUTHEVENT	Cardholder Authentication Events	Credit Card Payments	None, View, Edit, Full, Create

Permission ID	Permission Name	Feature	Valid Levels
LIST_CASE	Cases	Customer Support and Service	None, View, Create, Edit, Full
LIST_CASE_DUPLICATES	Duplicate Case Management		None, Full
LIST_CATEGORY	Expense Categories	Expense Reports	None, View, Create, Edit, Full
LIST_CERTIFICATES	Certificate Access	—	None, View, Create, Edit, Full
LIST_CHECKITEMAVAILABILITY	Check Item Availability	Advanced Inventory Management	None, View, Create, Edit, Full
LIST_CLASS	Classes	Classes	None, View, Create, Edit, Full
LIST_COLORTHEME	Color Themes	SuiteCommerce	None, View, Create, Edit, Full
LIST_COMMISSIONRULES	Employee Commission Schedules/Plans	Employee Commissions	None, View, Create, Edit, Full
LIST_COMPANY	Companies	Customer Relationship Management	None, View, Create, Edit, Full
LIST_COMPETITOR	Competitors	Sales Force Automation	None, View, Create, Edit, Full
LIST_COMPONENTWHEREUSEDINQUIRY	Component Where Used	Assembly Items	None, View, Create, Edit, Full
LIST_CONTACT	Contacts	Contacts	None, View, Create, Edit, Full
LIST_CONTACTROLE	Contact Roles	Sales Force Automation	None, View, Create, Edit, Full
LIST_CONVERTLEAD	Lead Conversion	—	View, None, Create, Full, Edit
LIST_COSTEDBOMINQUIRY	Costed Bill Of Materials Inquiry	Assembly Items	None, View, Create, Edit, Full
LIST_CRMGROUP	CRM Groups	Customer Relationship Management	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_CRMMESSAGE	Track Messages	Customer Relationship Management	None, View, Create, Edit, Full
LIST_CRMTEMPLATE	Marketing Template	Marketing Automation	None, View, Create, Edit, Full
LIST_CURRENCY	Currency	Multiple Currencies	None, View, Create, Edit, Full
LIST_CUSTJOB	Customers	Prospects and Contacts	None, View, Create, Edit, Full
LIST_CUSTPROFILE	Customer Profile	Customer Profile	None, View, Create, Edit, Full
LIST_CUSTRECORDENTRY	Custom Record Entries	Custom Records	None, View, Create, Edit, Full
LIST_DEPARTMENT	Departments	Departments	None, View, Create, Edit, Full
LIST_DISTRIBUTIONNETWORK	Distribution Network	Advanced Inventory Management	None, View, Create, Edit, Full
LIST_EARLIEST_AVAILABILITY	Earliest Availability	Supply Allocation	None, View, Create, Edit, Full
LIST_EMAILTEMPLATE	Email Template	Email Template	None, View, Create, Edit, Full
LIST_EMPLOYEE	Employees	Employees	None, View, Create, Edit, Full
LIST_EMPLOYEE_ACCESS	Employee Access Tab	Advanced Employee Permissions	None, View, Create, Edit, Full
LIST_EMPLOYEE_ ADMINISTRATION	Employee Administration	Advanced Employee Permissions	None, View, Create, Edit, Full
LIST_EMPLOYEE_ CONFIDENTIAL	Employee Confidential	Advanced Employee Permissions	None, View, Create, Edit, Full
LIST_EMPLOYEE_PUBLIC	Employee Public	Advanced Employee Permissions	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_EMPLOYEE_RECORD	Employee Record	—	View, Create, Edit, Full
LIST_EMPLOYEE_SELF	Employee Self	Advanced Employee Permissions	View, Create, Edit, Full
LIST_EMPLOYEECHANGETYPE	Employee Change Request Type	—	None, View, Create, Edit, Full
LIST_EMPLOYEECHANGEREASON	Employee Change Reason	Effective Dating	None, View, Create, Edit, Full
LIST_EMPLOYEECHANGEREQUEST	Employee Change Request	Employee Change Requests	None, View, Create, Edit, Full
LIST_EMPLOYEECHANGETYPE	Employee Change Request Type	Employee Change Requests	None, View, Create, Edit, Full
LIST_EMPLOYEEEFFECTIVEDATING	Employee Effective Dating	Effective Dating	None, View, Create, Edit, Full
LIST_EMPLOYEESEPARATION	Termination Reasons	Termination Reason Tracking	None, View, Create, Edit, Full
LIST_EMPLOYEESSN	Employee Social Security Numbers	Employees	None, View, Full
LIST_ENTITY_DUPLICATES	Duplicate Entity Management	Duplicate Detection & Merge	None, View, Full
LIST_EVENT	Events	Events	None, View, Create, Edit, Full
LIST_EXPENSEAMORTIZATIONRULE	Expense Amortization Rule	—	None, View, Create, Edit, Full
LIST_EXPENSEPLAN	Expense Amortization Plan	—	None, View, Create, Edit, Full
LIST_EXPORT	Export Lists	Search Result Export	None, Create
LIST_FAIRVALUEDIMENSION	Fair Value Dimension	Revenue Recognition	None, View, Create, Edit, Full
LIST_FAIRVALUEFORMULA	Fair Value Formula	Revenue Recognition	None, View, Create, Edit, Full
LIST_FAIRVALUEPRICE	Fair Value Price	Revenue Recognition	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_FAXMESSAGE	Fax Messages	Communications	None, View, Create, Edit, Full
LIST_FAXTEMPLATE	Fax Template	Mail Merge	None, View, Create, Edit, Full
LIST_FILECABINET	Documents and Files	File Cabinet	None, View, Create, Edit, Full
LIST_FIND	Perform Search	Order Management	None, View, Full
LIST_FINHISTORY	Financial History		None, View, Create, Edit, Full
LIST_FISCALCALENDAR	Fiscal Calendars	Accounting	None, View, Create, Edit, Full
LIST_GENERAL_TOKEN	General Token	Payment Instruments	None, View, Create, Edit, Full
LIST_GENERICRESOURCE	Generic Resources	Project Management	None, View, Create, Edit, Full
LIST_GLLINESAUDITLOG	Custom GL Lines Plug-in Audit Log	Custom GL Lines Plug-in	None, View, Create, Edit, Full
LIST_GLLINESAUDITLOGSEG	Custom GL Lines Plug-in Audit Log (Segments)	Custom GL Lines Plug-in	None, View, Create, Edit, Full
LIST_GLOBALINVTRELATIONSHIP	Global Inventory Relationship	—	None, View, Create, Edit, Full
LIST_GOVERNMENTISSUEDIDTYPE	Government-Issued ID Types	Advanced Government-Issued ID Tracking	None, View, Create, Edit, Full
LIST_HCMJOB	HCMJob Management	—	None, View, Create, Edit, Full
LIST_HCMPOSITION	Positions	—	None, View, Create, Edit, Full
LIST_HISTORY	Notes Tab	Communications	None, View, Create, Edit, Full
LIST_IMPORTED_EMPLOYEE_EXPENSE	Imported Employee Expenses	Expense Reports	View, None

Permission ID	Permission Name	Feature	Valid Levels
LIST_INBOUNDSHIPMENT	Inbound Shipment	Inbound Shipment Management	None, View, Create, Edit, Full
LIST_INFOCATEGORY	Store Content Categories	Web Store	None, View, Create, Edit, Full
LIST_INFOITEM	Store Content Items	SuiteCommerce	None, View, Create, Edit, Full
LIST_INFOITEMFORM	Publish Forms	SuiteCommerce	None, View, Create, Edit, Full
LIST_INTEGRAPP	Integration Applications	SuiteTalk	None, View, Create, Edit, Full
LIST_INTERNALPUBLISH	Internal Publisher		None, View, Create, Edit, Full
LIST_INVCOSTTEMPLATE	Inventory Cost Template	Multi-Book Accounting	None, View, Create, Edit, Full
LIST_INVENTORYSTATUS	Inventory Status	Inventory Management	None, View, Create, Edit, Full
LIST_ISSUE	Issues	Issue Management	None, View, Create, Edit, Full
LIST_ITEM	Items	Item Record Management	None, View, Create, Edit, Full
LIST_ITEM_COLLECTION	Item Collection	—	None, View, Create, Edit, Full
LIST_ITEMDEMANDPLAN	Item Demand Plan	Advanced Inventory Management	None, View, Create, Edit, Full
LIST_ITEMREVENUECATEGORY	Item Revenue Category	Revenue Recognition	None, View, Create, Edit, Full
LIST_ITEMPROCESSFAMILY	Item Process Family	Warehouse Management	None, View, Create, Edit, Full
LIST_ITEMPROCESSGROUP	Item Process Group	Warehouse Management	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_ITEM_REVISION	Item Revisions	Item Record Management	None, View, Create, Edit, Full
LIST_ITEMSUPPLYPLAN	Item Supply Plan	Advanced Inventory Management	None, View, Create, Edit, Full
LIST_ITEMTEMPLATE	Item Templates		None, View, Create, Edit, Full
LIST_JOB	Jobs	Projects	None, View, Create, Edit, Full
LIST_JOBREQUISITION	HCMJob Requisitions	Job Requisitions	None, View, Create, Edit, Full
LIST_KEYS	Key access	—	None, View, Create, Edit, Full
LIST_KUDOS	Kudos	Kudos	None, View, Create, Edit, Full
LIST_KNOWLEDGEBASE	Knowledge Base	Knowledge Base	None, View, Create, Edit, Full
LIST_LABORCOSTING	Labor Costing	Labor Costing	View, None, Full, Create, Edit
LIST_LOCATION	Locations	Locations	None, View, Create, Edit, Full
LIST_MAILMERGE	Mail Merge	Mail Merge	None, View, Create, Edit, Full
LIST_MAILMESSAGE	Letter Messages	Communications	None, View, Create, Edit, Full
LIST_MAILTEMPLATE	Letter Template	Mail Merge	None, View, Create, Edit, Full
LIST_MASSUPDATES	Mass Updates	Mass Update	None, View, Create, Edit, Full
LIST_MATERIALREQUIREMENTSPLAN	Material Requirements Planning	Material Requirements Planning	View, None, Full, Create, Edit

Permission ID	Permission Name	Feature	Valid Levels
LIST_MEDIAITEMFOLDER	Media Folders	File Cabinet	None, View, Create, Edit, Full
LIST_MEMDOC	Memorized Transactions	Transactions	None, View, Create, Edit, Full
LIST_MFGCOSTTEMPLATE	Manufacturing Cost Template	Inventory Management	None, View, Create, Edit, Full
LIST_MFGROUTING	Manufacturing Routing	Inventory Management	None, View, Create, Edit, Full
LIST_NEWSITEM	News Items	—	None, View, Create, Edit, Full
LIST_NOTIFICATION	Notifications	—	None, View, Create, Edit, Full
LIST_ONBOARDING_PLAN	Onboarding Plan	—	None, View, Create, Edit, Full
LIST_ORDER_REALLOCATION	Commit Orders	Advanced Inventory Management	None, View, Create, Edit, Full
LIST_ORDERMANAGEDASHBOARD	Order Management Dashboard	Sales Channel Allocation	None, View, Create, Edit, Full
LIST_ORGANIZATIONVALUE	Organizational Value	Kudos	None, View, Create, Edit, Full
LIST_OTHERNAME	Other Names	Sales Force Automation	None, View, Create, Edit, Full
LIST_OUTBOUNDREQUEST	Outbound Request	—	None, View, Create, Edit, Full
LIST_OVERTIMEPOLICY	Overtime Policies		None, View, Create, Edit, Full
LIST_PARTNER	Partners	Partner Relationship Management	None, View, Create, Edit, Full
LIST_PARTNERCOMMISSNRULES	Partner Commission Schedules/Plans	Partner Commissions/Royalties	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_PAYCHECK	Paychecks	Payroll	None, View, Create, Edit, Full
LIST_PAYMENT_CARD	Payment Card	Payment Instruments	None, View, Create, Edit, Full
LIST_PAYMENT_CARD_TOKEN	Payment Card Token	Payment Instruments	None, View, Create, Edit, Full
LIST_PAYMENT_INSTRUMENTS	Payment Instruments	Payment Instruments	None, View, Create, Edit, Full
LIST_PAYMETH	Payment Methods	Order Management	None, View, Create, Edit, Full
LIST_PAYROLLITEM	Payroll Items	Payroll	None, View, Create, Edit, Full
LIST_PDFMESSAGE	PDF Messages	Communications	None, View, Create, Edit, Full
LIST_PDFTEMPLATE	PDF Template	Mail Merge	None, View, Create, Edit, Full
LIST_PHASEDPROCESS	Phased Processes		None, View, Create, Edit, Full
LIST_PICKSTRATEGY	Pick Strategy	Warehouse Management	None, View, Create, Edit, Full
LIST_PICKTASK	Pick Task	Warehouse Management	None, View, Create, Edit, Full
LIST_PLANNEDREVENUE	Planned Revenue	Revenue Recognition	None, View, Create, Edit, Full
LIST_PLANNEDSTANDARDCOST	Planned Standard Cost	Item Record Management	None, View, Create, Edit, Full
LIST_PRESCATEGORY	Presentation Categories	SuiteCommerce	None, View, Create, Edit, Full
LIST_PRICEBOOK	Price Books	—	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_PRICEPLAN	Price Plans	—	None, View, Create, Edit, Full
LIST_PROJECT_BUDGET	Project Budget	Advanced Project Budgets	None, View, Create, Edit, Full
LIST_PROJECTREVENUERULE	Project Revenue Rules	Project Management	None, View, Create, Edit, Full
LIST_PROJECTTASK	Project Tasks	Project Management	None, View, Create, Edit, Full
LIST_PROJECTTEMPLATE	Project Templates	Project Management	None, View, Create, Edit, Full
LIST_PROMOTIONCODE	Promotion	Sales Force Automation	None, View, Create, Edit, Full
LIST_PUBLISHSEARCH	Publish Search	Publishing Search Results	None, View, Create, Edit, Full
LIST_QUANTITYPRICINGSCHEDULE	Quantity pricing Schedules	Quantity Pricing	None, View, Create, Edit, Full
LIST_REALLOCATE_ORDER_ITEM	Reallocate Order Item	Supply Allocation	View, None, Full, Create, Edit
LIST_RECOGNITIONEVENTTYPE	Custom Recognition Event Type	Advanced Revenue Recognition Or Advanced Expense Management	None, View, Create, Edit, Full
LIST_RECORDCUSTFIELD	Record Custom Field		None, View, Create, Edit, Full
LIST_RELATEDITEMS	Related Items	Web Site	None, View, Create, Edit, Full
LIST_RESOURCE	Resource	Project Management	None, View, Create, Edit, Full
LIST_RESOURCEGROUP	Resource Groups	Project Management	None, View, Create, Edit, Full
LIST_REVENUEELEMENT	Revenue Element	Revenue Recognition	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_REVENUEPLAN	Revenue Recognition Plan	Revenue Recognition	None, View, Create, Edit, Full
LIST_REVENUERECOGNITIONRULE	Revenue Recognition Rule	Revenue Recognition	None, View, Create, Edit, Full
LIST_REVRECSCHEDULE	Revenue Recognition Schedules	Revenue Recognition	None, View, Create, Edit, Full
LIST_REVRECTREATMENT	Recognition Treatment	—	None, View, Create, Edit, Full
LIST_REVRECTREATMENTRULE	Recognition Treatment Rule	—	None, View, Create, Edit, Full
LIST_REVRECFIELDMAPPING	Revenue Recognition Field Mapping	Revenue Recognition	None, View, Create, Edit, Full
LIST_REVRECVSOE	Revenue Management VSOE	VSOE	None, View, Create, Edit, Full
LIST_RSRCALLOCATION	Resource Allocations	Project Management	None, View, Create, Edit, Full
LIST_RSRCALLOCATIONAPPRV	Resource Allocation Approval	Project Management	None, View, Create, Edit, Full
LIST_RSSFEED	Publish RSS Feeds		None, View, Create, Edit, Full
LIST_SALESCAMPAIGN	Sales Campaigns	Sales Campaigns	None, View, Create, Edit, Full
LIST_SALESROLE	Sales Roles	Team Selling	None, View, Create, Edit, Full
LIST_SCSNAPSHOT	Supply Chain Snapshot List	—	None, View, Create, Edit, Full
LIST_SENTEMAIL	Sent Email	Sent Email List	None, View, Create, Edit, Full
LIST_SHIPITEM	Shipping Items	Accounting	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_SHIPPARTPACKAGE	Shipping Partner Package	Order Management	None, View, Create, Edit, Full
LIST_SHIPPARTREGISTRATION	Shipping Partner Registration	Order Management	None, View, Create, Edit, Full
LIST_SHIPPARTSHIPMENT	Shipping Partner Shipment	Order Management	None, View, Create, Edit, Full
LIST_SHORTCUT	Shortcuts		None, View, Create, Edit, Full
LIST_SCHEDULEMASSUPDATES	Schedule Mass Updates		None, View, Create, Edit, Full
LIST_SITEEMAILTEMPLATE	Web Store Email Template	SuiteCommerce	None, View, Create, Edit, Full
LIST_STANDARDCOSTVERSION	Standard Cost Version	Item Record Management	None, View, Create, Edit, Full
LIST_STORECATEGORY	Store Categories	Web Store	None, View, Create, Edit, Full
LIST_STOREITEMLISTLA	Item/Category Layouts	Web Store	None, View, Create, Edit, Full
LIST_STORETAB	Store Tabs	Web Store	None, View, Create, Edit, Full
LIST_SUBSCRIPTION	Subscriptions	Advanced Subscription Billing	None, View, Create, Edit, Full
LIST_SUBSCRIPTIONCHANGEORDER	Subscription Change Orders	Advanced Subscription Billing	None, View, Create, Edit, Full
LIST_SUBSCRIPTIONPLAN	Subscription Plan	Advanced Subscription Billing	None, View, Create, Edit, Full
LIST_SUBSIDIARY	Subsidiaries	Subsidiaries	None, View, Create, Edit, Full
LIST_SUPPLY_REALLOCATION	Allocate Orders	—	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_SYSTEMEMAILTEMPLATE	System Email Template	Customer Relationship Management	None, View, Create, Edit, Full
LIST_TALENT_ ADMINISTRATION	Talent Administration	—	View, Full
LIST_TASK	Tasks	Customer Relationship Management	None, View, Create, Edit, Full
LIST_TAXDETAILSTAB	Tax Details Tab	Tax Overhauling	None, View, Edit, Full
LIST_TAXENGINESELECTION	Subsidiary Tax Registrations Tab	Tax Overhauling	None, View, Edit, Full
LIST_TAXITEM	Tax Records	Accounting	None, View, Create, Edit, Full
LIST_TAXSCHEDULE	Tax Schedules	Advanced Taxes	None, View, Create, Edit, Full
LIST_TEGATAACCOUNT	Tegata Accounts	Accounting	None, View, Create, Edit, Full
LIST_TEMPLATE_CATEGORY	Template Categories	Email Marketing Campaigns	None, View, Create, Edit, Full
LIST_TIMECODE	Time Codes	Overtime	None, View, Create, Edit, Full
LIST_TIMEOFFADMIN	Time-Off Administration	Time-Off Management	None, View, Create, Edit, Full
LIST_TRANNUMBERAUDITLOG	Access to transaction numbering audit log	Transactions	None, View, Create, Edit, Full
LIST_UNDELIVEREDEMAIL	Undelivered Emails	—	None, View
LIST_UNIT	Units	Accounting	None, View, Create, Edit, Full
LIST_UPSELL	Upsell Assistant	Upsell Manager	None, View, Create, Edit, Full
LIST_UPSELLWIZARD	Upsell Wizard	Upsell Manager	None, View, Create, Edit, Full
LIST_USAGE	Usage	Advanced Subscription Billing	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
LIST_VENDOR	Vendors	Accounting	None, View, Create, Edit, Full
LIST_WBS	Work Breakdown Structure	Advanced Project Budgets	None, View, Create, Edit, Full
LIST_WEBSITE	Website (External) publisher	Web Site	None, View, Create, Edit, Full
LIST_WORKASSIGNMENT	Work Assignments	—	View, None, Full, Create, Edit
LIST_WORKCALENDAR	Work Calendar	Project Management	None, View, Create, Edit, Full
LIST_WORKPLACE	Workplaces	Payroll	None, View, Create, Edit, Full
LIST_ZONE	Zone	Warehouse Management	None, View, Create, Edit, Full
REGT_ACCTPAY	Accounts Payable Register	A/P	None, View, Create, Edit, Full
REGT_ACCTREC	Accounts Receivable Register	A/R	None, View, Create, Edit, Full
REGT_BANK	Bank Account Registers	Accounting	None, View, Create, Edit, Full
REGT_COGS	Cost of Goods Sold Registers	Accounting	None, View, Create, Edit, Full
REGT_CREDCARD	Credit Card Registers	Accounting	None, View, Create, Edit, Full
REGT_DEFEREXPENSE	Deferred Expense Registers	Amortization	None, View, Create, Edit, Full
REGT_DEFERREVENUE	Deferred Revenue Registers	Revenue Recognition	None, View, Create, Edit, Full
REGT_EQUITY	Equity Registers	Accounting	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
REGT_EXPENSE	Expense Registers	Accounting	None, View, Create, Edit, Full
REGT_FIXEDASSET	Fixed Asset Registers	Accounting	None, View, Create, Edit, Full
REGT_INCOME	Income Registers	Accounting	None, View, Create, Edit, Full
REGT_LONGTERMLIAB	Long Term Liability Registers	Accounting	None, View, Create, Edit, Full
REGT_NONPOSTING	Non Posting Registers	Accounting	None, View, Create, Edit, Full
REGT_OTHASSET	Other Asset Registers	Accounting	None, View, Create, Edit, Full
REGT_OTHCURRASSET	Other Current Asset Registers	Accounting	None, View, Create, Edit, Full
REGT_OTHCURRLIAB	Other Current Liability Registers	Accounting	None, View, Create, Edit, Full
REGT_OTHEXPENSE	Other Expense Registers	Accounting	None, View, Create, Edit, Full
REGT_OTHINCOME	Other Income Registers	Accounting	None, View, Create, Edit, Full
REGT_PAYROLL	Run Payroll	Payroll	None, View, Create, Edit, Full
REGT_STAT	Statistical Account Registers	Statistical Accounting	None, View, Create, Edit, Full
REGT_UNBILLEDREC	Unbilled Receivable Registers	Revenue Commitments	None, View, Create, Edit, Full
REPO_1099	Form 1099 - Federal Miscellaneous Income	A/P	None, View, Create, Edit, Full
REPO_940	Form 940 - Employer's Annual Federal Unemployment Tax Return	Payroll	None, View

Permission ID	Permission Name	Feature	Valid Levels
REPO_941	Form 941 - Employer's Quarterly Federal Tax Return	Payroll	None, View
REPO_ACCOUNTDETAIL	Account Detail	Accounting	None, View
REPO_AMORTIZATION	Amortization Reports	Amortization	None, View
REPO_ANALYTICS	SuiteAnalytics Workbook	SuiteAnalytics Workbook	None, Edit
REPO_AP	Accounts Payable	A/P	None, View
REPO_AR	Accounts Receivable	A/R	None, View
REPO_AUTHPARTNERCOMMISSION	Partner Authorized Commission Reports	Partner Commissions/Royalties	None, View
REPO_BALANCESHEET	Balance Sheet	Accounting	None, View
REPO_BOOKINGS	Sales Order Reports	Sales Force Automation	None, View
REPO_BUDGET	Budget	Accounting	None, View
REPO_CASHFLOW	Cash Flow Statement	Accounting	None, View
REPO_COMMISSION	Commission Reports	Employee Commissions	None, View
REPO_CUSTOMIZATION	Report Customization	Report Customization	None, View
REPO_DEFERREDEXPENSE	Deferred Expense Reports	—	None, View
REPO_FINANCIALS	Financial Statements	Accounting	None, View
REPO_GL	General Ledger	Accounting	None, View
REPO_GRANT_ACCESS	Granting access to Reports		None, View, Create, Edit, Full
REPO_GSTSUMMARY	GST Summary Report		None, View
REPO_INTEGRATION	Integration	SuiteTalk	None, View
REPO_INVENTORY	Inventory	Inventory	None, View
REPO_ISSUE	Issue Reports	Issue Management	None, View
REPO_MARKETING	Marketing Campaign Reports	Marketing Automation	None, View
REPO_NONPOSTING	Sales Order Transaction Report	Order Management	None, View
REPO_PANDL	Income Statement	Accounting	None, View
REPO_PARTNERCOMMISSION	Partner Commission Reports	Partner Commissions/Royalties	None, View
REPO_PAYCHECKDETAIL	Payroll Check Register	Payroll	None, View
REPO_PAYROLL	Payroll Summary & Detail Reports	Payroll	None, View
REPO_PAYROLLHIDEFINEMPINFO	Hide Employee Information on Financial Reports	Payroll	None, View
REPO_PAYROLLHOURSEARNING	Payroll Hours & Earnings	Payroll	None, View

Permission ID	Permission Name	Feature	Valid Levels
REPO_PAYROLLJOURNAL	Payroll Journal Report	Payroll	None, View
REPO_PAYROLLLIAB	Payroll Liability Report	Payroll	None, View
REPO_PAYROLLSTATEWITHHOLD	Payroll State Withholding	Payroll	None, View
REPO_PAYROLLW2	Form W-2 - Wage and Tax Statement	Payroll	None, View, Create, Edit, Full
REPO_PERIODENDFINANCIALS	Period End Financial Statements	—	None, View
REPO_PROJECT_ACCOUNTING	Project Accounting	Accounting	None, View
REPO_PSTSUMMARY	PST Summary Report		None, View
REPO_PURCHASEORDER	Purchase Order Reports	Purchase Orders	None, View
REPO_PURCHASES	Purchases	Accounting	None, View
REPO_QUOTA	Quota Reports	Sales Force Automation	None, View
REPO_RECONCILE	Reconcile Reporting	Accounting	None, View
REPO_REMINDEREMPLOYEE	Employee Reminders	Employees	None, View
REPO_RETURNAUTH	Return Authorization Reports	Return Authorizations	None, View
REPO_REVREC	Revenue Recognition Reports	Revenue Recognition	None, View
REPO_RSRCALLOCATION	Resource Allocation Reports	Project Management	None, View
REPO_SALES	Sales	Sales Force Automation	None, View
REPO_SALESORDER	Sales Order Fulfillment Reports	Sales Orders	None, View
REPO_SALES_PARTNER	Sales By Partner	Partner Relationship Management	None, View
REPO_SALES_PROMO	Sales By Promotion Code	Sales Force Automation	None, View
REPO_SALES_PROMO	Sales By Promotion	Sales Force Automation	None, View
REPO_SCHEDULE	Report Scheduling	Reports	None, Full
REPO_SNAPSHOTCASE	Support Case Snapshot/ Reminders	Support	None, View
REPO_SNAPSHOTLEAD	Lead Snapshot/Reminders	Business	None, View
REPO_SFA	Sales Force Automation	Customer Relationship Management	None, View
REPO_SUPPORT	Support	Customer Support and Service	None, View
REPO_TAX	Tax	Accounting	None, View
REPO_TAXREPORTS	Tax Reports	Accounting	None, View
REPO_TIME	Time Tracking	Time Tracking	None, View
REPO_TRAN	Transaction Detail	Transactions	None, View

Permission ID	Permission Name	Feature	Valid Levels
REPO_TRIALBALANCE	Trial Balance	Accounting	None, View
REPO_UNBILLED	Accounts Receivable Un-Billed	Accounting	None, View
REPO_W4	Form W4 - Employee's Withholding Allowance Certificate	Payroll	None, View
REPO_WEBSITE	Web Site Report	Web Site	None, View
REPO_WEBSTORE	Web Store Report	Web Store	None, View
REPO_WORKFORCEANALYTICS	Workforce Analytics	Workforce Analytics	View
TRAN_ADJUSTMENTJOURNAL	Currency Adjustment Journal	Accounting	None, View, Create, Edit, Full
TRAN_ALLOCSCHEDULE	Create Allocation Schedules	Expense Allocation	None, View, Create, Edit, Full
TRAN_AMENDW4	Amend W-4	Employees	None, View, Create, Edit, Full
TRAN_APPROVECOMMISSN	Employee Commission Transaction Approval	Employee Commissions	None, View, Create, Edit, Full
TRAN_APPROVEDD	Approve Direct Deposit	Direct Deposit	None, View, Create, Edit, Full
TRAN_APPROVEEFT	Approve EFT	Electronic Funds Transfer	None, View, Create, Edit, Full
TRAN_APPROVEPARTNERCOMM	Partner Commission Transaction Approval	Partner Commissions/Royalties	None, View, Create, Edit, Full
TRAN_AUDIT	Audit Trail	Transactions	None, View, Create, Edit, Full
TRAN_AUTO_CASH	Automated Cash Application	Accounting	None, Full
TRAN_BALANCEOVERVIEW	Balance Overview	Intercompany Framework	None, View
TRAN_BALJRNAL	Balancing Journals	—	None, View, Create, Full
TRAN_BINTRNFR	Bin Transfer	Bin Management	None, View, Create, Edit, Full
TRAN_BINWKSHT	Bin Putaway Worksheet	Bin Management	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_BLANKORD	Blanket Purchase Order	Purchasing and Receiving	None, View, Create, Edit, Full
TRAN_BLANKORDAPPRV	Blanket Purchase Order Approval	Purchasing and Receiving	None, View, Create, Edit, Full
TRAN_BUDGET	Set Up Budgets	Accounting	None, View, Create, Edit, Full
TRAN_BUILD	Build Assemblies	Assembly Items	None, View, Create, Edit, Full
TRAN_CARDCHRG	Credit Card	Accounting	None, View, Create, Edit, Full
TRAN_CARDHOLDERAUTHENTICATION	Cardholder Authentication	Credit Card Payments	None, View, Edit, Full, Create
TRAN_CARDHOLDERAUTHEVENT	Cardholder Authentication Event	Credit Card Payments	None, View, Edit, Full, Create
TRAN_CARDRFND	Credit Card Refund	Accounting	None, View, Create, Edit, Full
TRAN_CASHRFND	Cash Sale Refund	Accounting	None, View, Create, Edit, Full
TRAN_CASHSALE	Cash Sale	Accounting	None, View, Create, Edit, Full
TRAN_CHARGE	Charge	Project Management	None, View, Create, Edit, Full
TRAN_CHARGERULE	Charge Rule	Project Management	None, View, Create, Edit, Full
TRAN_CHECK	Check	Accounting	None, View, Create, Edit, Full
TRAN_CLEARHOLD	Override Payment Hold	Order Management	None, View, Create, Edit, Full
TRAN_COMMISSN	Employee Commission Transaction	Employee Commissions	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_COMMITPAYROLL	Commit Payroll	Payroll	None, View, Create, Edit, Full
TRAN_COPY_BUDGET	Copy Budgets	Accounting	None, View, Create, Edit, Full
TRAN_CREATEINVCOUNT	Create Inventory Counts	Inventory Count	None, View, Create, Edit, Full
TRAN_CUSTAUTH	Customer Payment Authorization	Customer Payment Authorizations	None, View, Create, Edit, Full
TRAN_CUSTCHRG	Statement Charge	A/R	None, View, Create, Edit, Full
TRAN_CUSTCRED	Credit Memo	A/R	None, View, Create, Edit, Full
TRAN_CUSTDEP	Customer Deposit	A/R	None, View, Create, Edit, Full
TRAN_CUSTINVC	Invoice	A/R	None, View, Create, Edit, Full
TRAN_CUSTINVCAPPRV	Invoice Approval	Order Management	None, View, Create, Edit, Full
TRAN_CUSTPYMT	Customer Payment	A/R	None, View, Create, Edit, Full
TRAN_CUSTRFND	Customer Refund	A/R	None, View, Create, Edit, Full
TRAN_DEPAPPL	Deposit Application	A/R	None, View, Create, Edit, Full
TRAN_DEPOSIT	Deposit	Accounting	None, View, Create, Edit, Full
TRAN_EDITBANKINGINFO	Personal Banking Information	—	None, View, Full
TRAN_EDITPROFILE	Edit Profile	Employees	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_ESTIMATE	Estimate	Estimates	None, View, Create, Edit, Full
TRAN_ESTIMATEDCOSTOVERRIDE	Override Estimated Cost on Transactions		None, Full
TRAN_EXPREPT	Expense Report	Expense Reports	None, View, Create, Edit, Full
TRAN_FFTREQ	Fulfillment Request	Order Management	None, View, Create, Edit, Full
TRAN_FINCHRG	Finance Charge	A/R	None, View, Create, Edit, Full
TRAN_FIND	Find Transaction	Transactions	None, View, Create, Edit, Full
TRAN_FORECAST	Edit Forecast	Sales Force Automation	None, View, Create, Edit, Full
TRAN_FXREVAL	Currency Revaluation	Multiple Currencies	None, View, Create, Edit, Full
TRAN_GST_REFUND	Process GST Refund	Accounting	None, View, Create, Edit, Full
TRAN_IMPORTOLBFILE	Import Online Banking (QIF) File	Accounting	None, View, Create, Edit, Full
TRAN_INTERCOADJ	Intercompany Adjustments	Accounting	None, View, Create, Edit, Full
TRAN_INVADJST	Adjust Inventory	Inventory	None, View, Create, Edit, Full
TRAN_INVCOUNT	Count Inventory	Advanced Inventory Management	None, View, Create, Edit, Full
TRAN_INVDISTR	Distribute Inventory	Multi-Location Inventory	None, View, Create, Edit, Full
TRAN_INVREVAL	Revalue Inventory Cost	Inventory	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_INVTRNFR	Transfer Inventory	Multi-Location Inventory	None, View, Create, Edit, Full
TRAN_INVWKSHT	Adjust Inventory Worksheet	Inventory	None, View, Create, Edit, Full
TRAN_ITEMRCPT	Item Receipt	Advanced Receiving	None, View, Create, Edit, Full
TRAN_ITEMSHIP	Item Fulfillment	Advanced Shipping	None, View, Create, Edit, Full
TRAN_JOURNAL	Make Journal Entry	Accounting	None, View, Create, Edit, Full
TRAN_JOURNALAPPRV	Journal Approval	Accounting	None, View, Create, Edit, Full
TRAN_LIABPYMT	Payroll Liability Payments	Payroll	None, View, Create, Edit, Full
TRAN_MANAGEPAYROLL	Manage Payroll	Payroll	None, View, Create, Edit, Full
TRAN_MATCHING_RULES	Matching Rules for Online Banking	Accounting	None, View, Create, Edit, Full
TRAN_MGRFORECAST	Edit Manager Forecast	Sales Force Automation	None, View, Create, Edit, Full
TRAN_NETTINGSETTLEMENTAPPRV	Netting Settlement Approval	Intercompany Framework	None, Edit
TRAN_NETTSTLM	Netting Settlement	Intercompany Framework	None, View, Create, Edit, Full
TRAN_OPENBAL	Enter Opening Balances	Accounting	None, View, Create, Edit, Full
TRAN_OPPRTNTY	Opportunity	Opportunities	None, View, Create, Edit, Full
TRAN_ORDERRESERVATION	Order Reservation	—	None, View, Edit, Full, Create

Permission ID	Permission Name	Feature	Valid Levels
TRAN_ORDRESVAPPRV	Approve Order Reservation		None, View, Create, Edit, Full
TRAN_OWNTRNSF	Ownership Transfer	—	None, View, Create, Edit, Full
TRAN_PARTNERCOMMISSN	Partner Commission Transaction	Partner Commissions/Royalties	None, View, Create, Edit, Full
TRAN_PAYCHECK	Individual Paycheck	Payroll	None, View, Create, Edit, Full
TRAN_PAYMENTAUDIT	Access Payment Audit Log	Order Management	None, View, Create, Edit, Full
TRAN_PAYMENTEVENT	View Payment Events	Credit Card Payments	None, View, Create, Edit, Full
TRAN_PAYMENTRESULTPREVIEW	View Payment Result Previews	Credit Card Payments	None, View, Edit, Full, Create
TRAN_PAYROLLRUN	Process Payroll	Payroll	None, View, Create, Edit, Full
TRAN_PCHKJRNL	Paycheck Journal	Employees	None, View, Create, Edit, Full
TRAN_PEJRNL	Period End Journals	—	None, View, Create, Edit, Full
TRAN_POSTPERIODS	Posting Period on Transactions	Accounting	None, View, Create, Edit, Full
TRAN_POSTVENDORBILLVARIANCE	Post Vendor Bill Variances	Vendors	None, View, Create, Edit, Full
TRAN_PRICELIST	Generate Price Lists	Item Record Management	None, View, Create, Edit, Full
TRAN_PRINTSHIPMENTDOCS	Print Shipment Documents	Shipping Partners	None, View, Create, Edit, Full
TRAN_PROJECT_IC_CHARGE_REQUEST	Project Intercompany Cross Charge Request	Project Intercompany Cross Charge Request	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_PURCHCON	Purchase Contract	Vendors	None, View, Create, Edit, Full
TRAN_PURCHCONAPPRV	Purchase Contract Approval	Vendors	None, View, Create, Edit, Full
TRAN_PURCHORD	Purchase Order	Purchase Orders	None, View, Create, Edit, Full
TRAN_PURCHORDBILL	Bill Purchase Orders	Advanced Receiving	None, View, Create, Edit, Full
TRAN_PURCHORDRECEIVE	Receive Order	Purchase Orders	None, View, Create, Edit, Full
TRAN_PURCHREQ	Requisition	Purchasing and Receiving	None, View, Create, Edit, Full
TRAN_PURCHREQAPPRV	Requisition Approval	Purchasing and Receiving	None, View, Create, Edit, Full
TRAN_QUOTA	Establish Quotas	Sales Force Automation	None, View, Create, Edit, Full
TRAN_RECOG_GIFTCERT_INCOME	Recognize Gift Certificate Income	Accounting	None, View, Create, Edit, Full
TRAN_RECONCILE	Reconcile	Accounting	None, View, Create, Edit, Full
TRAN_REVARRNG	Revenue Arrangement	Revenue Recognition	None, View, Create, Edit, Full
TRAN_REVARRNGAPPRV	Revenue Arrangement Approval	Revenue Recognition	None, View, Create, Edit, Full
TRAN_REVCOMM	Revenue Commitment	Revenue Commitments	None, View, Create, Edit, Full
TRAN_REVCOMRV	Revenue Commitment Reversal	Revenue Commitments	None, View, Create, Edit, Full
TRAN_REVCONTR	Revenue Contracts	Revenue Recognition	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_RFQ	Request For Quote	Purchasing and Receiving	None, View, Create, Edit, Full
TRAN_RTNAUTH	Return Authorization	Return Authorizations	None, View, Create, Edit, Full
TRAN_RTNAUTHAPPRV	Return Auth. Approval	Return Authorizations	None, View, Create, Edit, Full
TRAN_RTNAUTHCREDIT	Refund Returns	Order Management	None, View, Create, Edit, Full
TRAN_RTNAUTHRECEIVE	Receive Returns	Order Management	None, View, Create, Edit, Full
TRAN_RTNAUTHREVERSEREVCOMMIT	Generate Revenue Commitment Reversals	Revenue Commitments	None, View, Create, Edit, Full
TRAN_SALESORD	Sales Order	Sales Orders	None, View, Create, Edit, Full
TRAN_SALESORDAPPRV	Sales Order Approval	Sales Orders	None, View, Create, Edit, Full
TRAN_SALESORDCOMMITREVENUE	Generate Revenue Commitment	Revenue Commitments	None, View, Create, Edit, Full
TRAN_SALESORDFULFILL	Fulfill Orders	Order Management	None, View, Create, Edit, Full
TRAN_SALESORDINVOICE	Invoice Sales Orders	Advanced Shipping	None, View, Create, Edit, Full
TRAN_SALESORDREVENUECONTRACT	Generate Single Order Revenue Contracts	Revenue Recognition	None, View, Create, Edit, Full
TRAN_STATEMENT	Generate Statements	A/R	None, View, Create, Edit, Full
TRAN_STATCHNG	Inventory Status Change	Inventory Status	None, View, Create, Edit, Full
TRAN_STATUSDD	Direct Deposit Status	Direct Deposit	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_STATUSEFT	EFT Status	Electronic Funds Transfer	None, View, Create, Edit, Full
TRAN_STPICKUP	Store Pickup Fulfillment	SuiteCommerce InStore	None, View, Create, Edit, Full
TRAN_SYSJRNL	System Journal	Accounting	None, View, Create, Edit, Full
TRAN_TAXLIAB	Pay Tax Liability	Accounting	None, View, Create, Edit, Full
TRAN_TAXPYMT	Pay Sales Tax	Accounting	None, View, Create, Edit, Full
TRAN_TEGPYBL	Tegata Payable	Accounting	None, View, Create, Edit, Full
TRAN_TEGRCVBL	Tegata Receivable	Accounting	None, View, Create, Edit, Full
TRAN_TIMEBILL	Track Time	Time Tracking	None, View, Create, Edit, Full
TRAN_TIMECALC	Calculate Time	Time Tracking	None, View, Create, Edit, Full
TRAN_TIMEPOST	Post Time	Project Management	None, View, Create, Edit, Full
TRAN_TIMER	Timer	Time Tracking	None, View, Create, Edit, Full
TRAN_TRANSFER	Transfer Funds	Accounting	None, View, Create, Edit, Full
TRAN_TRNFRORD	Transfer Order	Inventory Management	None, View, Create, Edit, Full
TRAN_TRNFRORDAPPRV	Transfer Order Approval	Inventory Management	None, View, Create, Edit, Full
TRAN_UNBUILD	Unbuild Assemblies	Assembly Items	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_VENDAUTH	Vendor Return Authorization	Vendor Return Authorizations	None, View, Create, Edit, Full
TRAN_VENDAUTHAPPRV	Vendor Return Auth. Approval	Vendor Return Authorizations	None, View, Create, Edit, Full
TRAN_VENDAUTHCREDIT	Credit Returns	Vendor Returns	None, View, Create, Edit, Full
TRAN_VENDAUTHRETURN	Vendor Returns	Vendor Return Authorizations	None, View, Create, Edit, Full
TRAN_VENDBILL	Bills	A/P	None, View, Create, Edit, Full
TRAN_VENDBILLAPPRV	Vendor Bill Approval	Vendor Bills	None, View, Create, Edit, Full
TRAN_VENDCRED	Enter Vendor Credits	A/P	None, View, Create, Edit, Full
TRAN_VENDPYMT	Pay Bills	A/P	None, View, Create, Edit, Full
TRAN_VENDPYMTAPPRV	Vendor Payment Approval	Vendor Bills	None, View, Create, Edit, Full
TRAN_VENDRFQ	Vendor Request For Quote	Purchasing and Receiving	None, View, Create, Edit, Full
TRAN_VPREPAPP	Vendor Prepayment Application	—	None, View, Create, Edit, Full
TRAN_VPREP	Vendor Prepayment	—	None, View, Create, Edit, Full
TRAN_WAVE	Wave	—	None, View, Create, Edit, Full
TRAN_WOCLOSE	Work Order Close	Inventory Management	None, View, Create, Edit, Full
TRAN_WOCOMPL	Work Order Completion	Inventory Management	None, View, Create, Edit, Full

Permission ID	Permission Name	Feature	Valid Levels
TRAN_WOISSUE	Work Order Issue	Inventory Management	None, View, Create, Edit, Full
TRAN_WORKORD	Work Order	Work Orders	None, View, Create, Edit, Full
TRAN_WORKORDBUILD	Build Work Orders	Work Orders	None, View, Create, Edit, Full
TRAN_WORKORDCLOSE	Close Work Orders	Inventory Management	None, View, Create, Edit, Full
TRAN_WORKORDCOMPLETE	Enter Completions	Inventory Management	None, View, Create, Edit, Full
TRAN_WORKORDISSUE	Issue Components	Inventory Management	None, View, Create, Edit, Full
TRAN_WORKORDMARKBUILT	Mark Work Orders Built	Inventory Management	None, View, Create, Edit, Full
TRAN_WORKORDMARKFIRMED	Mark Work Orders Firmed	Inventory Management	None, View, Create, Edit, Full
TRAN_WORKORDMARKRELEASED	Mark Work Orders Released	Inventory Management	None, View, Create, Edit, Full
TRAN_XCHGJRNL	Cross Charge Journal	Intercompany Framework	None, View
TRAN_YTDADJST	Enter Year-To-Date Payroll Adjustments	Payroll	None, View, Create, Edit, Full

Feature Names and IDs

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following table provides the internal IDs and feature names for all NetSuite features. You can use the feature ID to see if a particular feature is enabled in your account by calling runtime.isFeatureInEffect(options) in the N/runtime Module.

Feature Internal ID	Feature Name
ACCOUNTING	Accounting
ACCOUNTINGPERIODS	Accounting Periods
ACTIVITYCODES	Activity Codes
ADDONS	Add-On Items

Feature Internal ID	Feature Name
ADVANCEDBILLOFMATERIALS	Advanced Bill of Materials
ADVANCEDEMPLOYEEPERMISSIONS	Advanced Employee Permissions
ADVANCEDJOBS	Project Management
ADVANCEDNUMBERINGSEQUENCES	Advanced Numbering
ADVANCEDPRINTING	Advanced PDF/HTML Templates
ADVANCEDPROCUREMENTAPPROVALS	Advanced Procurement Approvals
ADVANCEDPROJECTACCOUNTING	Advanced Project Profitability
ADVANCEDPROMOTIONS	Advanced Promotions
ADVANCEDREVENUERECOGNITION	Advanced Revenue Management
ADVANCEDREVENUERECOGNITIONAPP	Advanced Revenue Recognition SuiteApp
ADVANCEDSITECUST	Advanced Site Customization
ADVANCEDSITEMANAGEMENT	Site Management Tools
ADVBILLING	Advanced Billing
ADVBINSERIALLOTMGMT	Advanced Bin/Numbered Inventory Management
ADVFORECASTING	Advanced Forecasting
ADVINVENTORYMGMT	Advanced Inventory Management
ADVPARTNERACCESS	Advanced Partner Access
ADVRECEIVING	Advanced Receiving
ADVSHIPPING	Advanced Shipping
ADVSUBSCRIPTIONBILLING	Advanced Subscription Billing
ADVTAXENGINE	Advanced Taxes
ADVWEBREPORTS	Advanced Web Reports
ADVWEBSEARCH	Advanced Web Search
ALTSALESADVFORECAST	ALT_SALES Advanced Forecasting
ALTSALESAMOUNT	Alternate Sales Amount
AMORTIZATION	Amortization
APPROVALROUTING	Approval Routing
ARMREVENUEALLOCATION	Advanced Revenue Management (Revenue Allocation)
ASSEMBLIES	Assembly Items
ASYNCCUSTOMER	Asynchronous AfterSubmit Customer Processing
ASYNCSALESORDER	Asynchronous AfterSubmit Sales Order Processing
AUTOAPPLYPROMOTIONS	Auto-Apply Promotions

Feature Internal ID	Feature Name
AUTOLOCATIONASSIGNMENT	Automatic Location Assignment
AVAILABLETOPROMISE	Available To Promise
BALANCING_SEGMENTS	Balancing Segments
BARCODES	Bar Coding and Item Labels
BILLCAPTURE	Bill Capture
BILLINGACCOUNTS	Billing Accounts
BILLINGCLASSES	Per-Employee Billing Rates
BILLINGRATECARDS	Billing Rate Cards
BILLINGWORKCENTER	Billing Operations
BILLSCOSTS	Bill Costs To Customers
BINMANAGEMENT	Bin Management
BLANKETPURCHASEORDERS	Blanket Purchase Orders
BOXNET	Box Document Management
CAMPAIGNASSISTANT	Campaign Assistant
CAMPAIGNSUBSCRIPTIONS	Subscription Categories
CCTRACKING	Credit Card Payments
CENTRALIZEDPURCHASINGBILLING	Centralized Purchasing and Billing
CHARGEBASEDBILLING	Charge-Based Billing
CHECKOUTSUBDOMAIN	Customizable Checkout Subdomains
CLASSES	Classes
COMMERCECATEGORIES	Commerce Categories
COMMERCESEARCHANALYTICS	Commerce Search Analytics
COMMISSIONONCUSTOMFIELDS	Commission on Custom Fields
COMMISSIONS	Employee Commissions
COMPENSATIONTRACKING	Compensation Tracking
CONSOLPAYMENTS	Consolidated Payments
CREATESUITEBUNDLES	Create bundles with SuiteBundler
CRM	Customer Relationship Management
CRMTIME	Time Tracking for CRM
CRM_TEMPLATE_CATEGORIES	CRM Template Categories
CROSSSUBSIDIARYFULFILLMENT	Intercompany Cross-Subsidiary Fulfillment
CUSTOMCODE	Client SuiteScript

Feature Internal ID	Feature Name
CUSTOMERACCESS	Customer Access
CUSTOMGLLINES	Custom GL Lines
CUSTOMRECORDS	Custom Records
CUSTOMSEGMENTS	Custom Segments
CUSTOMTRANSACTIONS	Custom Transactions
DEPARTMENTS	Departments
DISTRIBUTIONRESOURCEPLANNING	Distribution Resource Planning
DOCUMENTPUBLISHING	Document Publishing
DOCUMENTS	File Cabinet
DOWNLOADITEMS	Sell Downloadable Files
DROPSHIPMENTS	Drop Shipments & Special Orders
DUPLICATES	Duplicate Detection & Merge
DYNALLOCATION	Dynamic Allocation
EFFECTIVEDATING	Effective Dating
EMAILINTEGRATION	Capture Email Replies
EMPLOYEECENTERPUBLISHING	Employee Center Dashboard Publishing
EMPLOYEECHANGEREQUESTS	Employee Change Requests
EMPPERMS	Global Permissions
ENHANCEDINVENTORYLOCATION	Advanced Item Location Configuration
ENHANCEDPREMIERPAYROLL	Enhanced Premier Payroll
ESCALATIONRULES	Automated Case Escalation
ESTIMATES	Estimates
EXPENSEALLOCATION	Expense Allocation
EXPREPORTS	Expense Reports
EXTCRM	Online Forms
EXTREMELIST	Inline Editing
EXTSTORE	External Catalog Site (WSDK)
FCADVANCEDSECURITY	File Cabinet Advanced Security
FCEXPENSE	Enhanced File Security – Employee Expense Report Folders
FCEXPENSEMIGRATECONTROLLER	Expense Migration Scheduled From Start
FULFILLMENTREQUEST	Fulfillment Request
FXRATETYPE	Currency Exchange Rate Types

Feature Internal ID	Feature Name
FXRATEUPDATES	Currency Exchange Rate Integration
GAINLOSSACCTMAPPING	Foreign currency variance mapping
GIFTCERTIFICATES	Gift Certificates
GLAUDITNUMBERING	GL Audit Numbering
GRIDORDERMANAGEMENT	Grid Order Management
GROSSPROFIT	Gross Profit
GROUPAVERAGECOSTING	Group Average Costing
HELPDESK	Help Desk
HISTORICALMETRICS	Historical Metrics
HRANALYSIS	Workforce Analytics
I18NTAXREPORTS	International Tax Reports
IC_FRAMEWORK_OR_AIM	IC Framework OR Intercompany Auto Elimination
INBOUNDCASEEMAIL	Email Case Capture
INBOUNDSHIPMENT	Inbound Shipment Management
INSTALLMENTS	Installments
INTELLIGENTRECOMMENDATIONS	Intelligent Recommendations
INTERCOMPANYAUTODROPSHIP	Automated Intercompany Drop Ship
INTERCOMPANYAUTOELIMINATION	Automated Intercompany Management
INTERCOMPANYELIMINATIONENGINE	IC Framework OR IC Netting OR Advanced Inter. Elimination
INTERCOMPANYFRAMEWORK	Intercompany Framework
INTERCOMPANYTIMEEXPENSE	Intercompany Time and Expense
INTERNATIONALPHONENUMBERS	Phone Number Formatting
INTRANET	Intranet
INTRANSITPAYMENTS	In-Transit Payments
INVENTORY	Inventory
INVENTORYCOUNT	Inventory Count
INVENTORYSTATUS	Inventory Status
INVOICEGROUP	Invoice Groups
IPADDRESSRULES	IP Address Rules
ISSUEDB	Issue Management
ITEMDEMANDPLANNING	Demand Planning
ITEMOPTIONS	Item Options

Feature Internal ID	Feature Name
JOBCOSTING	Job Costing and Project Budgeting
JOBMANAGEMENT	Job Management
JOBREQUISITION	Job Requisitions
JOBS	Projects
KILLINBOUNDSSO	Disable Inbound Single Sign-on
KILLSHA1FORTBA	Disable HMAC-SHA1 for Token-based Authentication
KNOWLEDGEBASE	Knowledge Base
KPIREPORTS	KPI Scorecards
KUDOS	Kudos
LANDEDCOST	Landed Cost
LEADMANAGEMENT	Lead Conversion
LOCATIONS	Locations
LOTNUMBEREDINVENTORY	Lot Tracking
MAILMERGE	Mail Merge
MARKETING	Marketing Automation
MATERIALREQUIREMENTSPLANNING	Material Requirements Planning
MATRIXITEMS	Matrix Items
MERCHANDISEHIERARCHY	Merchandise Hierarchy
MFGROUTING	Manufacturing Routing and Work Center
MFGWORKINPROCESS	Manufacturing Work In Process
MOBILEPUSHNTF	Mobile Push Notification
MOSS	EU One Stop Shop
MULTIBOOKMULTICURR	Multi-Book Accounting and Multiple Currencies
MULTICURRENCY	Multiple Currencies
MULTICURRENCYCUSTOMER	Multi-Currency Customers
MULTICURRENCYMERGE	Multi Currency Merge
MULTICURRENCYVENDOR	Multi-Currency Vendors
MULTILANGUAGE	Multi-Language
MULTILOCINVT	Multi-Location Inventory
MULTIPARTNER	Multi-Partner Management
MULTIPLEBUDGETS	Multiple Budgets
MULTIPLECALENDARS	Multiple Calendars

Feature Internal ID	Feature Name
MULTISHIPTO	Multiple Shipping Routes
MULTISUBSIDIARYCUSTOMER	Multi Subsidiary Customer
MULTIVENDOR	Multiple Vendors
MULTPRICE	Multiple Prices
NETSUITEAPPROVALSWORKFLOW	NetSuite Approvals Workflow
NOTALTSALESAMOUNT	Not ALT_SALES Amount
NOTMULTIPARTNER	Not Multipartner
NOTTEAMSELLING	Not Team Selling
NSASOIDCPROVIDER	NetSuite as OIDC Provider
OAUTH2	OAuth 2.0 Authentication
OIDC	OpenID Connect (OIDC) Single Sign-on
ONLINEORDERING	Online Ordering
OPENIDSSO	OpenID Single Sign-on
OPPORTUNITIES	Opportunities
OTHERSUBLISTFIELDS	Other Sublist Fields
OUTSOURCEDMFG	Outsourced Manufacturing
PARTNERACCESS	Partner Access
PARTNERCOMMISSIONS	Partner Commissions/Royalties
PAYABLES	A/P
PAYCHECKJOURNAL	Paycheck Journal
PAYMENTINSTRUMENTS	Payment Instruments
PAYMENTLINK	Payment Link
PAYPALINTEGRATION	PayPal Integration
PAYROLL	Payroll
PAYROLLSERVICE	Payroll Service
PERFORMANCEMANAGEMENT	Performance Management
PERIODENDJOURNALENTRIES	Period End Journal Entries
PERSONALIZED_CATALOG_VIEWS	Personalized Catalog Views
PICKPACKSHIP	Pick, Pack and Ship
PI_REMOVAL	Remove Personal Information
PLANNEDWORK	Planned Work
PRM	Partner Relationship Management

Feature Internal ID	Feature Name
PROJECTTASKMANAGER	Project Task Manager
PROMOCODES	Promotion Codes
PURCHASECARDDATA	Send Purchase Card Data
PURCHASECONTRACTS	Purchase Contracts
PURCHASEORDERS	Purchase Orders
PURCHASEREQS	Purchase Requests
QUANTITYPRICING	Quantity Pricing
RECEIVABLES	A/R
REQUIREDDEPOSITWORKFLOW	Required Deposit Workflow
REQUISITIONS	Requisitions
RESOURCEALLOCATIONAPPROVAL	Resource Allocation Approval Workflow
RESOURCEALLOCATIONCHART	Resource Allocation Chart
RESOURCEALLOCATIONS	Resource Allocations
RESOURCESKILLSETS	Resource Skill Sets
RESTWEBSERVICES	REST Web Services
RETURNAUTHS	Return Authorizations
REVENUECOMMITMENTS	Revenue Commitments
REVENUERECOGNITION	Revenue Recognition
REVRECSALESORDERFORECASTING	Sales Order Revenue Forecasting
REVRECVSOE	VSOE
RFQ	Request For Quote
RULEBASEDRECOGNITIONTREATMENT	Rule-Based Recognition Treatment
SALESCAMPAIGNS	Sales Campaigns
SALESCHANNELALLOCATION	Sales Channel Allocation
SALESORDERS	Sales Orders
SAMLSSO	SAML Single Sign-on
SDFCOPYTOACCOUNT	Copy To Account
SERIALIZEDINVENTORY	Serialized Inventory
SERVERSIDESCRIPTING	Server SuiteScript
SERVICEPRINTEDCHECKS	Service Printed Checks and Stubs
SERVICEPRINTEDW2S	Service Printed W-2s and 1099s
SFA	Sales Force Automation

Feature Internal ID	Feature Name
SFA_AND_NOTASA	SFA And Not ALT_SALES Amount
SHIPPINGLABELS	Shipping Label Integration
SITEBUILDER	Site Builder (Website)
SITEBUILDER_STORE	Site Builder (Web Store)
SITELOCATIONALIASES	Descriptive URLs
SOFTDESCRIPTORS	Credit Card Soft Descriptors
STACKABLEPROMOTIONS	SuitePromotions
STANDARDCOSTING	Standard Costing
STATACCOUNTING	Statistical Accounts
STOREPICKUP	Store Pickup
SUBSCRIPTIONBILLING	Subscription Billing
SUITE_OAX_CONNECTOR	NetSuite Analytics Warehouse
SUITEANALYTICSCONNECT	SuiteAnalytics Connect
SUITEAPPCONTROLCENTER	SuiteApp Control Center
SUITEAPPDEVELOPMENTFRAMEWORK	SuiteCloud Development Framework
SUITECOMMERCE	SuiteCommerce
SUITECOMMERCE_ADVANCED	SuiteCommerce Advanced
SUITECOMMERCE_IN_STORE	SuiteCommerce In-Store
SUITECOMMERCE_MY_ACCOUNT	SuiteCommerce MyAccount
SUITECUBE_ENTERPRISE	Cached Data in Datasets
SUITESIGNON	SuiteSignOn
SUITETAXDATARECORDS	SuiteTax Data Records
SUITETAXENGINE	SuiteTax Engine
SUITETAXENGINEINDIA	India Localization SuiteTax Engine
SUITETAXREPORTS	SuiteTax Reports
SUITETAXREPORTSINDIA	India Localization SuiteTax Reports
SUPPLTAXCALC	Supplementary Tax Calculation
SUPPLYALLOCATION	Supply Allocation
SUPPLYCHAINCONTROLTOWER	Supply Chain Control Tower
SUPPLYCHAINMANAGEMENT	Supply Chain Management
SUPPLYCHAINPREDICTEDRISKS	Supply Chain Predicted Risks
SUPPORT	Customer Support and Service

Feature Internal ID	Feature Name
TABLEAU	Tableau Workbook Export
TAXAUDITFILES	Tax Audit Files
TBA	Token-based Authentication
TEAMSELLING	Team Selling
TELEPHONY	Telephony Integration
TIMEBASEDPRICING	Time-Based Pricing
TIMEBASEDPRICINGSUITEAPP	Subscription Billing Enhanced UI SuiteApp
TIMETRACKING	Time Tracking
TRANDELETIONREASONCODE	Use Deletion Reason
UNITSOFMEASURE	Multiple Units of Measure
UPSELL	Upsell Manager
URLCOMPONENTALIASES	URL Component Aliases
USR	SuiteAnalytics Workbook
VENDORACCESS	Vendor Access
VENDORPREPAYMENTS	Vendor Prepayments
VENDORRETURNAUTHS	Vendor Return Authorizations
WARRANTYANDREPAIRSMANAGEMENT	Warranty and Repairs Management
WBS	Advanced Project Budgets
WEBAPPLICATIONS	SuiteScript Server Pages
WEBDUPLICATEEMAILMANAGEMENT	Web Site Duplicate Email Management
WEBHOSTING	Host HTML Files
WEBSERVICESEXTERNAL	SOAP Web Services
WEBSITE	Web Site
WEEKLYTIMESHEETS	Weekly Timesheets
WEEKLYTIMESHEETSNEWUI	New Weekly Timesheets Interface
WITHHOLDINGTAX	Withholding Tax
WMSSYSTEM	Warehouse Management
WORKFLOW	SuiteFlow
WORKORDERS	Work Orders

Preference Names and IDs

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following tables list the internal IDs for all NetSuite preference configuration pages that support SuiteScript.

To interact with a configuration page, load the page using config.load(options). After the page loads, you can get or set configuration values using the returned record.Record object. Additionally, you can use User.getPreference(options) to get the values for **General Preferences** and **Accounting Preferences**.

NetSuite configuration preference IDs are grouped into the following categories:

- General Preferences
- Company Information
- User Preferences
- Accounting Preferences
- Accounting Periods
- Tax Setup
- Tax Periods

General Preferences

These are the account preferences that can be found by going to Setup > Company > General Preferences.

The internal ID for the General Preferences page is **companypreferences**. All preference internal IDs are case-insensitive.

Preference UI Label	Preference Internal ID
Date Format	DATEFORMAT
Long Date Format	LONGDATEFORMAT
Time Format	TIMEFORMAT
Number Format	NUMBERFORMAT
Negative Number Format	NEGATIVE_NUMBER_FORMAT
Phone Number Format	PHONEFORMAT
First Day of Week	FIRSTDAYOFWEEK
Search Sorting	SEARCHSORTING
Add Primary Contact to Bill To Address	CONTACTONBILLTO
Use Last Name First for Employees	LASTNAMEFIRST
Use Last Name First for Entities	LASTNAMEFIRSTENTITIES
Pre-Populate Contact Address	PREPOPULATECONTACTADDRESS
Show Employees as Contacts	SHOWEMPLOYEESASCONTACTS
Show Display Name with Item Codes	ITEMNUMBERING

Preference UI Label	Preference Internal ID
Password Policy	PASSWORD_POLICY
Minimum Password Length	MINPASSWORDLENGTH
Password Expiration in Days	PASSWORDEXPIREDAYS
Idle Session Timeout in Minutes	IDLE_SESSION_TIMEOUT
Internal Web Site	INTERNALWEBSITE
Allow Free-Form States in Addresses	FREEFORMSTATES
Use State Abbreviations in Addresses	ABBREVIATESTATES
Company Logo Folder	COMPANYLOGOFOLDER
Default Role for New Customers	CUSTOMERROLE
Assign Tasks to Partners	ASSIGNTASKSTOPARTNERS
Auto Name Customers	AUTONAMECUSTOMERS
Calendar System	CALENDARSYSTEM
Default Customer Type	CUSTOMERTYPE
Customer Center Welcome Message	CUSTOMERWELCOMEMESSAGE
Show Help Link In Customer Center	CUSTOMERSHOWHELPLINK
Horizontal Labels	HORIZONTALLABEL
Screen Font	FONT
Landing Page	CUSTOMLANDINGPAGE
Number of rows in List segments	LISTSEGMENTSIZE
Maximum entries in Dropdown	MAXDROPDOWNSIZE
Log System Notes on Update Only	LOGSYSTEMNOTESONUPDATEONLY
Show Quick Add Row on Lists	SHOWQUICKADD
Show List When Only One Results	SHOWLISTONRESULT
Default Customer Type	CUSTOMERTYPE
Default Lead Type	DEFAULTLEADTYPE
Default Vendor Type	DFLTVENDORTYPE
Default Partner Type	DFLTPARTNERTYPE
Auto Name Customers	AUTONAMECUSTOMERS
Auto Inactivate Contacts with Customers	INACTIVATE_CONTACTS
Show Individuals as Contacts	SHOWINDIVIDUALSASCONTACTS
Hide Attachment Folders	HIDEATTACHMENTFOLDERS
Assign Tasks to Partners	ASSIGNTASKSTOPARTNERS

Preference UI Label	Preference Internal ID
Email Employee on Approvals	EMAILEMPLOYEEONAPPROVAL
Maintenance Complete Email Notification	EMAILMAINTENANCECOMPLETE
Show Reports in Grid	REPORTGRID
Collapse VSOE Column By Default On Sales Transactions	COLLAPSEVSOEFIELDSET
Time Selectors Use Fiscal Calendars Based on First Month	TIMESELECTORSUSEFIRSTFISCALMONTH
Web Site Hosting Files Always Available	HOSTING_FILES_PUBLIC
Show Transaction Numbering Setup	SHOW_TRAN_NUMBERING_SETUP
Asynchronous Job Plan Recalculation	SHOW_TRAN_NUMBERING_SETUP
Preferred Subcustomer Form	SUBCUSTOMERFORM

Company Information

The Company Information preferences are available at Setup > Company > Company Information.

To interact with these preferences, use the `COMPANY_INFORMATION` value from the config.Type enum. All Company Information preference IDs are case-sensitive.

For details about working with the Company Information page in the UI, see the help topic Configuring Company Information.

Note: The Company Information page includes several preferences that are displayed as body fields. These fields are described in the following table.

Preference Label	Preference Internal ID
Company Name	companyname
Legal Name	legalname
Company Logo (Forms)	formlogo
Company Logo (Pages)	pagelogo
Display Logo Internally	displaylogointernally
Web Site	url
County/State/Province	state
Country	country
Return Email Address	email
Fax	fax
Currency	basecurrency
Employer Identification Number (EIN)	employerid
SSN or TIN (Social Security Number, Tax ID Number)	taxid

Preference Label	Preference Internal ID
First Fiscal Month	fiscalmonth
Time Zone	timezone
Account ID	companyid
Customer Center Login	customersurl
These three IDs represent text summaries of the data that exists on the three address subrecords.	mainaddress _text
	shippingaddress _text
	returnaddress _text

The Company Information page also includes gray boxes with the headings Address, Shipping Address, and Return Address. In the UI, you interact with these components by clicking the Edit link next to each box. Clicking any of these links displays a popup window that includes more fields.

To interact with these fields programmatically, you must use subrecord methods to access the fields associated with the Address, Shipping Address, and Return Address blocks. You must first instantiate the appropriate subrecord by using Record.getSubrecord(options). This method requires one argument: the ID of the subrecord. For the address subrecords on the Company Information page, use the following internal IDs:

- mainaddress
- shipaddress
- returnaddress

After you instantiate the subrecord, you can interact with any of the preferences shown in the following table. These preferences are handled as fields on each subrecord instance.

Preference Label	Preference Internal ID
Address1	addr1
Address2	addr2
Attention	attention
Addressee	addressee
Phone	addrphone
City	city
County/State/Province	state
Zip	zip
Country	country

For an example, see the help topic Retrieving a Body Field Address Subrecord Example. For general information about subrecords, see the help topic SuiteScript 2.x Scripting Subrecords.

User Preferences

These are the user preferences that can be found by going to Home > Set Preferences.

The internal ID for the user preferences page (which appears as the Set Preferences page in the UI) is **userpreferences**. All preference internal IDs are case-sensitive.

Be aware that the API for setting user preferences works the same as the UI in terms of setting a preference for a user's session or setting it permanently. In the UI if a user sets a preference, and the preference reverts back to a default setting on the user's next login, the same behavior is supported in SuiteScript.

Preference UI Label	Preference Internal ID
On the General tab	
Nickname	MESSAGE_NICKNAME
Signature	MESSAGE_SIGNATURE
Add Signature to Messages	MESSAGE_AUTOSIGNATURE
From Email Address	MESSAGE_EMAIL
Language	LANGUAGE
Search Sorting	SEARCHSORTING
Language of the Help Center	HELP_LANGUAGE
PDF Language	PDFLANGUAGE
Time Zone	TIMEZONE
First Day of the Week	FIRSTDAYOFWEEK
Date Format	DATEFORMAT
Long Date Format	LONGDATEFORMAT
Time Format	TIMEFORMAT
Number Format	NUMBERFORMAT
Negative Number Format	NEGATIVE_NUMBER_FORMAT
Phone Number Format	PHONEFORMAT
Auto Place Decimal	AUTOPLACE
CSV Column Delimiter	CSV_COLUMN_DELIMITER
CSV Decimal Delimiter	CSV_DECIMAL_DELIMITER
Use Multicurrency Expense Reports	USE_MC_ON_EXPREPT
Download PDF Files	DOWNLOADPDFS
Address Mapping Type	MAPTYPE
Show Internal IDs	EXPOSEIDS
Only Show Last Subaccount	ONLYSHOWLASTSUBACCT
Only Show Last Subentity	ONLYSHOWLASTSUBENT
Only Show Last Subitem	ONLYSHOWLASTSUBITEM
Submit Warnings	SUBMITWARNINGS

Preference UI Label	Preference Internal ID
Limit CC Field to Contacts & Employees	EMAILLIMITCC
Default Issue Email Notification	ISSUE_EMAIL_ME_WHEN
Notify Me Upon Issue Assignment	ISSUE_NOTIFY_UPON_ASSIGNMENT
Delay Loading of Sublists	DELAYLOADINGSUBLISTS
Number of Rows in List Segments	LISTSEGMENTSIZE
Maximum Entries in Dropdowns	MAXDROPDOWNSIZE
Type-Ahead on List Fields	TYPEAHEADSELECTS
Require Exact Match on Item Type-Ahead	ITEMEXACTMATCH
Show Quick Add Row on Lists	SHOWQUICKADD
Display Bounce Warning on Campaigns	CAMPAIGN_BOUNCE_WARNING
Prefer Native Select Fields Over NS Dropdowns in Internet Explorer	NATIVE_DROPDOWNS
Show App ID Field	SHOW_APPID_FIELD
Show ID Field on Sublists	SHOW_ID_FIELD
On the Appearance tab	
Color Theme	COLORTHEME
Screen Font	FONT
Compensate for Large Fonts	SYSTEMLARGEFONTS
Density Setting for Internet Explorer	MSIE_ZOOM_FACTOR
Register Look on Lists	REGISTERSTYLE
Only Show Field Boarders on Hover	SHOWFIELDBORDERONHOVER
Chart Theme	CHART_THEME
Chart Background	CHART_BACKGROUND
Landing Page	LANDINGPAGE
Show Portlet Hint	SHOWPORTLETHINT
Set Customer Dashboard As Default View On Customer Record	DASHBOARD_DEFAULT_VIEW_FOR_CUSTOMER
Limit Entry Forms to Two Columns	LIMITTOTWOCOLUMNS
Expand Tabs on Entry Forms	UNLAYEREDTABS
Enable Rich Text Editing	RICHTEXTEDITOR
Default Rich Text Editor Font	EDITORFONT
Default Rich Text Editor Font Size	EDITORFONTSIZE
Display Default Them With Optimal Color Contrast	ACCESSIBILITY_HIGH_CONTRAST

Preference UI Label	Preference Internal ID
On the Transactions tab	
Auto Fill Transactions	AUTOFILL
Alphabetize Items Regardless of Type	ALPHABETIZE_ITEMS
Duplicate Number Warnings	DUPLICATEWARNINGS
Inventory Level Warnings	STOCKWARNINGS
Customer Credit Limit Handling	CUSTCREDLIMHANDLING
Vendor Credit Limit Warnings	VENDCREDLIMWARNINGS
Print Using HTML	HTMLPRINTING
Transaction Email Attachment Format	TRANSACTION_ATTACHMENT_FORMAT
Horizontal Print Offset	HORZPRINTOFFSET
Vertical Print Offset	VERTPRINTOFFSET
On the Analytics tab	
Report by Period	REPORTBYPERIOD
Show Reports in Grid	REPORTGRID
Customize Font on Financial Reports	ENABLE_REPORT_FONT_CUSTOMIZATION
Print Company Logo	DISPLAYLOGO
Display Report Title on Screen	DISPLAYRPTTITLE
Display Report Description	DISPLAYRPTDESC
Calculate Forecasts as Weighted	FORECASTWEIGHTED
Default Bank Account	DEFAULT_BANKREG
Show Forecasts as Weighted	FORECASTWEIGHTED
Show List When Only One Result	SHOWLISTONERESULT
Quick Search Uses Keywords	KEYWORDSEARCH
Popup Search Uses Keywords	KEYWORDSEARCHPOPUP
Include Inactives in Global & Quick Search	SEARCHINACTIVES
Popup Auto Suggest	POPUPAUTOSUGGEST
Global Search Auto Suggest	SEARCHAUTOSUGGEST
Global Search Sort by Name/ID	GLOBALSEARCHSORTBYNAME
Global Search Customer Prefix Includes Leads and Prospects	GLOBALSEARCHCUPREFIX
PDF Page Orientation	REPORTPDFORIENTATION
PDF Font Size	REPORTPDFFONTSIZE
CSV Export Character Encoding	CSVEXPORTENCODING

Preference UI Label	Preference Internal ID
KPI Export Character Encoding	KPI_PERIOD_SPECIFIC_RATES
On the Activities tab	
Edit Activities from Calendar	EVENT_EDITFROMCALENDAR
Send Invitation Emails	EVENT_EMAILNOTIFICATION
Restrict Invitees to Employees	EVENT_INTERNALINVITEESONLY
Default Event Access Setting for New Events	EVENT_DEFAULTPUBLIC
Default Reminder Type	REMINDERTYPE
Default Reminder Time	REMINDERPERIOD
Play Audio with Popup Event Reminders	REMINDERPLAYWAVE
Default Priority for Tasks	DEFAULTTASKPRIORITY
Default New Tasks Public	TASK_DEFAULTPUBLIC
Default New Phone Calls Public	CALL_DEFAULTPUBLIC
Default Sync Category	DEFAULT_CONTACT_SYNC_CATEGORY
On the Alerts tab	
First Selection	EMAILALERT_AM
Second Selection	EMAILALERT_NOON
Third Selection	EMAILALERT_PM
Include links in HTML alerts	LINKS_EMAILALERT
Respect Quick Date Portlet Settings	USE_QUICKDATE_IN_ALERTS
E-Mail	EMAILALERT_EMAIL
Send an On-Demand Alert from this Role	ALERTONDEMAND
On the Restrict View tab	
Subsidiary	SUBSIDIARY
Include Sub-Subsidiaries	SUBSIDIARYSUBS
Department	DEPARTMENT
Include Sub-Departments	DEPARTMENTSUBS
Include Unassigned	DEPARTMENTUNASSIGNED
Location	LOCATION
Include Sub-Locations	LOCATIONSUBS
Include Unassigned	LOCATIONUNASSIGNED
Class	CLASS
Include Sub-Classes	CLASSSUBS

Preference UI Label	Preference Internal ID
Include Unassigned	CLASSUNASSIGNED
On the Telephony tab	
Telephony Option	TELEPHONY_OPTION
TAPI Device	TELEPHONYDEVICE
CTI URL	CTI_URL
Prefix to Dial Out	DIALOUTPREFIX

Accounting Preferences

These are the account preferences that can be found by going to Setup > Accounting > Accounting Preferences. All preference internal IDs are case-insensitive.

The internal ID for the Accounting Preferences page is **accountingpreferences**.

Preference UI Label	Preference Internal ID
On the General tab	
Use Account Numbers	ACCOUNTNUMBERS
Use Legal Name In Account	ACCOUNTLEGALNAME
Show All Transaction Types In Reconciliation	RECONCILIATIONALLTRANTYPES
Expand Account Lists	ASSETCOGSITEMACCTS
Cash Basis Reporting	CASHBASIS
Aging Reports Use	AGEFROM
Void Transactions Using Reversing Journals	REVERSALVOIDING
Set Reversal Variance Date Equal To The Reversing Journal Date When Voided Transaction Is In A Closed Period	SETREVERSINGVARIANCEDATETOREVJES
Use Journal Entry Summarization On Intercompany Elimination	ELIMINATION_JE_SUMMARIZATION
Require Approvals On Journal Entries	JOURNALAPPROVALS
Allow User Events On Bulk Journal Approval	BULK_JOURNAL_APPROVAL_EVENTS
Allow GL Custom Segment Deletion	CUSTOM_SEGMENT_DELETION
Enable Accounting Period Window	ALLOWFUTUREPERIODLOCK
Minimum Period Window Size	OPENCURFUTUREPERIODWINDOW
Allow Transaction Date Outside Of Posting Period	DATEPERIODMISMATCH
GL Audit Numbering Method	GLNUMBERINGBY

Preference UI Label	Preference Internal ID
Default Posting Period When Transaction Date In Closed Period	DEFAULTPERIODIFCLOSED
Create and Edit Inventory Transactions Dated In Closed Periods	INVT_TRANS_CLOSED_PERIODS
Allow Quick Close Of Accounting Periods	ALLOW_PERIODS_QUICK_CLOSE
Apply Payments Through Top-Level Customers Only	PAYMENTSONLYFROMTOPPARENT
Show Only Open Transactions On Statements	OPENONLYSTMTS
Open Transactions On Statements	OPENONLYTRANSACTIONS
Customer Credit Limit Handling	CUSTCREDLIMHANDLING
Customer Credit Limit Includes Orders	CUSTCREDLIMORDERS
Days Overdue For Warning / Hold	CREDLIMDAYS
Include Tax For Term Discounts	TERMDISCOUNTSINCLUDETAX
Include Shipping For Term Discounts	TERMDISCOUNTSINCLUDESHIPPING
Default Vendor Payments To Be Printed	VENDPYMTTOPRINT
Vendor Credit Limit Warnings	VENDCREDLIMWARNINGS
Vendor Credit Limit Includes Orders	VENDCREDLIMORDERS
Use In-Transit Vendor Payments By Default	DEFAULTVENDORPAYMENTTYPE
Vendor In-Transit Payment Account	VENDORITPACCOUNT
Vendor Prepayment Account	VENDORPREPAYMENTACCOUNT
Auto-Apply Vendor Prepayments	AUTO_APPLY_VENDOR_PREPAYMENTS
Allow Bill Consolidation Of Purchase Orders With Different Terms	ALLOWBILLCONSOLFORMULTITERMPO
Make Departments Mandatory	DEPTMANDATORY
Make Classes Mandatory	CLASSMANDATORY
Make Locations Mandatory	LOCMANDATORY
Allow Per-Line Departments	DEPTSPERLINE
Allow Per-Line Locations	LOCSPERLINE
Allow Per-Line Classes	CLASSESPERLINE
Always Allow Per-Line Classifications On Journals	CDLPERLINEONJE
Allow Non-Balancing Classifications On Journals	NONBALANCINGCDLONJE
Allow Empty Classifications On Journals	NULLCDLONJE
Allow Users to Modify Revenue Recognition Schedule	MODIFYREVRECTOTALAMOUNT
Prorate Revenue Recognition Dates For Partially Billed Sales Orders	PRORATEREVRECINVFROMSO

Preference UI Label	Preference Internal ID
Create Revenue Journals In GL	REVRECJOURNALENTRYSUMMARIZATION
Default Revenue Recognition Journal Date To	REVRECJOURNALDATEDEFAULT
Allow Users To Modify VSOE Values on Transactions	MODIFYVSOEVALSONTRAN
Use System Percentage Of Completion For Schedules	USESYSCALCPCT4REVREC
Adv Billing: Use Sales Order Amount	CALCPCTCOMPFROMSALESORDERAMT
Allow Revenue Commitments In Advance Of Fulfillment	UNSHIPPEDREVENUECOMMITMENTS
Allow Revenue Commitment Reversals In Advance Of Item Receipt	UNRECEIVEDREVENUECOMMITMENTS
Default Deferred Revenue Reclassification Account	DEFAULTDEFERREDREVENUEACCOUNT
Default Foreign Currency Adjustment Revenue Account	DEFAULTFXREVENUEACCOUNT
Allow Users To Modify Amortization Schedule	MODIFYAMORTOTALAMOUNT
Default Amortization Journal Date To	AMORJOURNALDATEDEFAULT
Default Amortization Journal Entry Form	AMORTIZATIONJOURNALENTRYFORM
Intercompany Time	INTERCOMPANYTIME
Intercompany Expenses	INTERCOMPANYEXPENSE
Intercompany Expenses	INTERCOMPANYEXPENSE
Enable Budget With Elimination Subsidiaries	BUDGETINCLUDEELIMSUB
Require Approvals on Journal Entries	JOURNALAPPROVALS
Rate Provider	EXCHANGE_RATE_PROVIDER
Use Triangulation Calculation By NetSuite	EXCHANGE_RATE_PROVIDER
Maximum number of MLI locations	MAXLOCATIONS
On the Items/Transactions tab	
Purchase Discount Account	PURCHDISCACCT
External Inventory In Transit Account	EXTINTRINVTACCT
Sales Discount Account	SALESDISCACCT
Default Expense Account	EXPENSEACCOUNT
Default Income Account	INCOMEACCOUNT
Default Receivables Account	ARACCOUNT
Default Cogs Account	COGSACCOUNT
Default Asset Account	ASSETACCOUNT
Default Payment Account	PAYMENTACCOUNT
Default Gain / Loss Account	GAINLOSSACCOUNT
Default Bill Quantity Variance Account	BILLPRICEVARIANCEACCT

Preference UI Label	Preference Internal ID
Default Vendor Return Variance Account	VENDRETURNVARIANCEACCOUNT
Default Customer Return Variance Account	CUSTOMERRETURNVARIANCEACCOUNT
Default Production Quantity Variance Account	PRODQTYVARIANCEACCT
Default Production Price Variance Account	PRODPRICEVARIANCEACCT
Default Purchase Price Variance Account	PURCHASEPRICEVARIANCEACCT
Default Inventory Count Account	INVCOUNTACCOUNT
Anyone Can Set Item Accounts	EDITITEMACCOUNTS
Default Dropship Expense Account	DROPSHIPEXPENSEACCOUNT
Consolidate Jobs on Sales Transactions	CONSOLINVOICES
Maximum # of Quantity-based Price Levels	QTYPRICECOUNT
Allow Quantity Discounts per Price Level on Schedules	QTYPRICESCHEDULEMULTDISCOUNTS
Include Reimbursements in Sales and Forecast Reports	FORECASTINCLUDES_REIMB_EXP
Include Shipping in Sales and Forecast Reports	FORECASTINCLUDES_SHIPPING
Transaction Types to Exclude from Forecast Reports	FORECASTTRANTYPES
Transaction Types to Exclude from Sales Reports	SALESTRANTYPES
Default Fixed Date Charge Rule Stage	FFCRFIXEDDATE
Default Milestone Charge Rule Stage	FFCRMILESTONE
Default Project Progress Charge Rule Stage	FFCRPROGRESS
Default Time-Based Rule Stage	TIMECR
Scan Individual Items	SINGLEITEMBARCODING
Centralize Purchasing In A Single Location	CENTRALIZEDPURCHASING
Days Before Lot Expiration Warning	LOTEXPIRATIONWARNING
Require Bins on All Transactions Except Item Receipts	REQUIREBINSONTRANS
Use Preferred Bin on Item Receipts	USEPREFERREDBINONITEMRCPT
Include Landed Cost In Last Purchase Price	INCLUDELANDEDCOSTINLPP
Inventory Costing Method	INVTCOSTMETHOD
Default Cost Category	DEFAULTCOSTCATEGORY
Average Cost Completion Unit Cost	AVGCOSTCOMPLETIONCOSTINGMETHOD
Use Intransit Value In Group Average Cost Calculations	USEINTRANSITINGACCALCULATIONS
Customers Can Pay Online	EXTERNALPAYMENTS
Get Authorization On Customer Center Sales Orders	AUTHORIZECUSTOMERCENTERORDERS
Use Card Security Code For Credit Card Transactions	CCSECURITYCODE

Preference UI Label	Preference Internal ID
Allow Adjusted Expiration Date To Improve Recurring Payments	ALLOWADJUSTEDEXPIRATIONDATE
Enable “Sale” Payment Operations On A Sales Order By Automatically Creating A Customer Deposit	CREATE_DEPOSIT_ON_SALES_ORDER_SALE
Use Strict Rules For The Selection Of Payment Processing Profiles	STRICT_SELECTION_OF_PPP
Preserve Transactions When Payment Is On Hold	PRESERVE_TRAN_ON_HOLD_PAYMENT
Duplicate Number Warnings	DUPLICATEWARNINGS
Sort Reconcile By	RECONSORTCOL
Recalculate Estimated Cost on Creation of Linked Transactions	USELATESTCOSTESTIMATE
Matrix Item Name/Number Separator	SKUSEPARATOR
Gift Certificate Auth Code Generation	GIFTCERTAUTHCODEGENERATION
Enforce Minimum Quantity On Return Authorizations	ENFORCE_MIN_QUANTITY_RET_AUTH
On the Order Management tab	
Default Sales Order Status	DEFSALESORDSTATUS
Require Re-approval on Edit of Sales Order	REAPPROVESOONEDIT
Send Email Confirmation when Sales Order Canceled	EMAILCANCELORDER
Default Location for Sales Orders	DEFAULTSALESORDERLOCATION
Default Commit Option On Sales Order	DEFAULTSALESORDERCOMMITOPTION
Default Commit Option On Transfer Order	DEFAULTTRANSFERORDERCOMMITOPTION
Item Commitment Transaction Ordering	ITEMCOMMITMENTTRANSACTIONORDER
Perform Item Commitment After Transaction Entry	AUTOMATICITEMCOMMITMENT
Always Print Kit Items on Picking Tickets	PICKINGTICKETKITITEMS
Show Non-Inventory Items on Picking Tickets and Packing Slips	PICKINGTICKETNONINVT
Show Uncommitted Items on Picking Tickets	PICKINGTICKETUNCOMMITTED
Name for Packed Status	NAMINGPACKED
Name for Picked Status	NAMINGPICKED
Name for Shipped Status	NAMINGSHIPPED
Show Additional Items on Packing Slips	SHOWADDLITEMSPACKSLIP
Show Drop Ship Items on Packing Slips	PACKINGSLIPDROPSHIP
Limit Status on Packing Slip Queue	PACKINGSLIPSTATUS
Fulfill Based on Commitment	FULFILLCOMMITTED

Preference UI Label	Preference Internal ID
Default Items to Zero Received/Fulfilled	DEFAULTUNFULFILLED
Allow Overage on Assembly Builds	OVERBUILDS
Filter Bulk Fulfillment Page by Location	BULKFULFILLOCFILTERING
Send Order Fulfilled Confirmation Emails	ORDFULFILLCONFEMAIL
Use Web Site Template for Fulfillment Emails	ORDFULFILLUSESTORETEMPLATES
Update Transaction Date Upon Fulfillment Status Change	UPDATEITEMSHIPDATEONSTATUSCHANGE
Show Unfulfilled Items on Invoices	SHOWUNSHIPPEDITEMS
Invoice In Advance Of Fulfillment	UNSHIPPEDINVOICES
Convert Absolute Discounts to Percentage When Billing	CONVERTABSOLUTEDISCOUNTS
Base Invoice Date on Billing Schedule Date	INCOICEUSESCHEDULEDATE
Drop Ship P.O. Form	DROPSHIPTEMPLATE
Automatically Email Drop Ship P.O.s	EMAILDROPSHIPPOS
Queue Drop Ship P.O.s for Printing	PRINTDROPSHIPPOS
Automatically Fax Drop Ship P.O.s	FAXDROPSHIPPOS
Include Committed Quantities	DROPSHIPINCLUDECOMMITTED
Limit Vendor List on Items	LIMITITEMVENDORS
Update Drop Ship Order Quantities Automatically Prior To Shipment	KEEPDROPSHIPQUANTITIESINSYNC
Drop Ship FulFillment Quantity Validation	FULFILLDROPSHIPORDERFROMINVENTORY
Allow Both Mark Shipped Fulfillments And Receipts On A Drop Shipment Line	DROPSHIPANDRECEIVEINTOINVDROPSHIPPO
Update Special Order Quantities Automatically Prior To Shipment	KEEPSPECIALORDERQUANTITIESINSYNC
Allow Expenses On Purchases	POEXPENSES
Default Location for Purchase Orders	DEFAULTPURCHASEORDERLOCATION
Maximum {Purchase} Lines To Consolidate	MAXPURCHASES
Allow Default Email On Purchase Orders Using SuiteFlow Approval Routing	ALLOW_DEFAULT_EMAIL_ON_PO
Default Number Of Requests For Quote Pricing Tiers	DEFAULT_RFQ_TIERS
Bill in Advance of Receipt	UNCRECEIVEDBILLS
Allow Overage on Item Receipts	OVERRECEIPTS
Default Receiving Exchange Rate	DEFAULTRECEIVINGEXCHANGERATE
Use Purchase Order Rate On Bills	USEPORATEONBILLS

Preference UI Label	Preference Internal ID
Landed Cost Allocation Per Line	LANDEDCOSTPERLINEDEFAULT
Default Return Auth. Status	DEFRTNAUTHSTATUS
Refund in Advance of Return	UNCRECEIVEDRTNAUTHS
Restock Returned Items	RESTOCKRETURNS
Write-Off Account for Returns	WRITEOFFACCOUNT
Default Vendor Return Auth. Status	DEFVENDAUTHSTATUS
Credit in Advance of Vendor Return	UNRETURNEDVENDAUTHS
Default Transfer Order Status	DEFTRNFRORDSTATUS
Generate Transfer Orders In Supply Planning	GENERATETRNFRORDINPLANNING
Use Item Cost As Transfer Cost	ITEMCOSTASTRNFRORDCOST
Default Transfer Order Incoterms	DEFAULT_TRNFRORD_INCOTERM
Default Vendor Bill Status	DEFVENDBILLSTATUS
On the Time & Expenses tab	
Show Jobs Only for Time and Expense Entry	TIMEEXPENSEJOBONLY
Automatically Notify Supervisor	AUTONOTIFYSUPV
Expenses Billable by Default	DEFAULTEXPENSEBILLABLE
Items Billable by Default	DEFAULTITEMSBILLABLE
Combine Detail Items on Expense Reports	COMBINEEXPENSEITEMS
Copy Expense Memos to Invoices	COPYEXPENSEMEMOS
Default Advance To Apply Account For Expense Reports	DEFAULT_ADVANCE_ACCT_FOR_EXPREPT
Foreign Amount Change	EXP_REPORT_FOREIGN_AMOUNT_CHANGE
On the Approval Routing tab	
Expense Reports	CUSTOMAPPROVALEXPENSE
Purchase Orders	CUSTOMAPPROVALPURCHORD
Vendor Bills	CUSTOMAPPROVALVENDORBILL
Requisitions	CUSTOMAPPROVALPURCHREQ
Invoices	CUSTOMAPPROVALCUSTINVC
Purchase Contracts	CUSTOMAPPROVALPURCHCON
Blanket Purchase Orders	CUSTOMAPPROVALBLANKORD
Journal Entries	CUSTOMAPPROVALJOURNAL

Accounting Periods

These are the account preferences that can be found by going to Setup > Accounting > Manage G/L > Manage Accounting Periods.

The internal ID for the Accounting Periods page is **accountingperiods**. All preference internal IDs are case-insensitive.

Preference UI Label	Preference Internal ID
First Fiscal Month	fiscalmonth
Fiscal Year End	fiscalyear
Period Format	periodstyle
Year in Period Name	periodnameyear
One-Day Year-End Adjustment Period	lastday

Manufacturing Preferences

These are the manufacturing preferences that can be found by going to Setup > Manufacturing > Manufacturing Preferences.

The internal ID for the Accounting Periods page is **manufacturingpreferences**. All preference internal IDs are case-insensitive.

Preference UI Label	Preference Internal ID
Default Unbuild Variance Account	UNBUILDVARIANCEACCOUNT
Default WIP Cost Variance Account	WIPVARIANCEACCT
Default Scrap Account	SCRAPACCT
Default WIP Account	WIPACCT
Average Cost Consumption Unit Cost	AVGCOSTSOMPLETIONSCOSTINGMETHOD
Allow Purchase Of Assembly Items	ALLOWASSEMBLYPURCHASE
Allow Editing of Legacy BOMs	ALLOWEDITINGOFMIGRATEDLEGACYBOMS
Default Scheduling Method	DEFAULTSCHEDULINGMETHOD
Default Commit Option On Work Order	DEFAULTWORKORDERCOMMITOPTION
Manufacturing Issue Based On Commitment	BUILDCOMMITTED
Create Work Orders In Supply Planning	CREATEWORKORDERINPLANNING
Default Allocation Strategy on WorkOrder	DEFAULTWORKORDERALLOCATIONSTRATEGY
Allow Overage On Work Order Transactions	OVERBUILDS
Default Work Order Status	DEFAULTWORKORDERSTATUS
Check Completed Quantity In Prior Operations During Operation Completion	VALIDATEPREDECESSORCOMPLETEDQTY

Preference UI Label	Preference Internal ID
Show Planned Capacity On Work Orders	SHOWPLANNEDCAPACITYONWORKORDERS
Automatically fill actual production start and end dates	FILL_ACTUAL_PRODUCTION_DATES

Tax Setup

The internal ID for the Set Up Taxes page is **taxpreferences**. All preference internal IDs are case-insensitive.

Preference IDs for fields on this page vary according to the country of the nexus. Field internal IDs are suffixed with the nexus country code. The format for these preferences is <field internal id><nexus country code>. Be aware that different fields are available for different nexuses.

The table below shows some scriptable tax preference fields for a US nexus. Fields are suffixed with **us**, for the US nexus. This table is provided for example purposes.

Preference UI Label	Preference Internal ID
	defaulttaxableus
	type
	chargoutofdistrictus
	perlinetaxesus
	storeordertaxationus
	enabletaxlookupus

Field IDs in your account may be suffixed with a different nexus country code. And different fields may be available. You may be able to look up field IDs in the user interface, by going to Setup > Accounting > Set Up Taxes, and clicking on a nexus.

- To make field IDs available, go to Home > Set Preferences and ensure that the Show Internal IDs box is checked on the General subtab, Defaults area.
- Find the field in the NetSuite user interface and click the field label to display the field level help text. The field ID is displayed in the popup.

Tax Periods

These are the tax preferences that can be found by going to Setup > Accounting > Taxes > Manage Tax Periods.

The internal ID for the Tax Periods page is **taxperiods**. All preference internal IDs are case-insensitive.

Preference UI Label	Preference Internal ID
First Fiscal Month	fiscalmonth
Fiscal Year End	fiscalyear
Period Format	periodstyle
Year in Period Name	periodnameyear

Task IDs

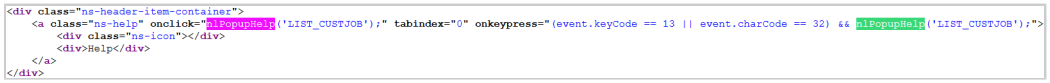
Applies to: SuiteScript 2.x | SuiteCloud Developer

Task IDs identify NetSuite pages. Each NetSuite page has a unique task ID.

You can reference task IDs when using the url.resolveTaskLink(options) and redirect.toTaskLink(options) methods. A task ID is available only if the related feature is enabled in your account.

To find a task ID for a NetSuite page:

1. Right-click in the NetSuite page, and select **View Page Source** from the context menu. Note that the exact command varies by browser.
2. Press Ctrl+F to open the search popup, and enter **NLPopupHelp** in the search field. Note that the search mechanism varies by browser.
3. Press Enter, use the down arrow, or click **Find Next** until `nlPopupHelp` is part of a line of code. The `nlPopupHelp` value is the task ID. In the following screenshot, `LIST_CUSTJOB` is the task ID for the NetSuite Customers page.



Button IDs

Applies to: SuiteScript 2.x | SuiteCloud Developer

The following table lists the internal IDs for standard NetSuite buttons that support SuiteScript.

When using SuiteScript to rename or hide buttons, use the Button.label and Button.isHidden properties of the serverWidget.Button object, respectively. On some records, certain buttons may appear as actions in the More Actions menu. You can still use the properties of serverWidget.Button to change the labels of these actions and to hide or show the actions in the menu. However, you cannot use SuiteScript to change the display of an inline button to an action in the More Actions menu. Similarly, you cannot use SuiteScript to display an action as an inline button. To change the display type of buttons and actions, you must use SuiteBuilder point-and-click customization. For more information, see the help topic Configuring Buttons and Actions.

Important: Customizing the Save, Edit, Cancel, Back, and Reset buttons is not supported in SuiteScript or in point-and-click customization.

Button UI Label	Button Internal ID
Add Items	addmatrix
Accept	accept
Accept Payment	acceptpayment
Apply	apply
Approve	approve
Approve Return	approvereturn
Authorize Return	return

Button UI Label	Button Internal ID
Auto Fill	autofill
Bill	bill
Bill Remaining	billremaining
Cancel Order	cancelorder
Cancel Return	cancelreturn
Clear Splits	clearsplits
Close	closeremaining
Convert	convertlead
Convert to Inventory	convertinvt
Convert to Lot Numbered Inventory	convertlot
Convert to Serialized Inventory	convertserial
Create Build	createbuild
Create Matrix	creatematrix
Credit	credit
Decline	decline
Delete	delete
Email	email
Fax	fax
Fulfill	process
Generate Price List	generatepricelist
Generate Statement	generatestatement
GL Impact	glimpact
Go To Register	gotoregister
Grab	grab
Make Copy	makecopy
Make Payment	payment
Make Standalone Copy	makestandalonecopy
Memorize	memorize
Merge	merge
New	new
New Event Field	neweventfield
Next Bill	nextbill

Button UI Label	Button Internal ID
Next Week	next
Prev Week	prev
Print	print
Print Bill of Materials	printbom
Print Label	printlabel
Print Labels	printlabels
Print Picking Ticket	printpicktick
Print Summary	depositsummary
Quick Accept	quickaccept
Recalc	recalc
Receive	receive
Refund	refund
Reject	reject
Renew	renewal
Reset	resetter
Revenue Commitment Reversal	revcomrv
Save As	submitas
Save & Bill	submitbill
Save & Convert	submitconvert
Save & Copy	submitcopy
Save & Edit	submitedit
Save & Email	saveemail
Save & Fulfill	submitfulfill
Save & New	submitnew
Save & Next	submitnext
Save & Print	saveprint
Save & Print BOM	saveprintbom
Save & Print Label	saveandprintlabel
Save & Refund	submitrefund
Save & Same	submitsame
Save Baseline	savebaseline
Search	search

Button UI Label	Button Internal ID
Show Activity	showactivity
Submit Invoice	submitinvoice
Tentative	tentative
Unbuild	createunbuild
Update Matrix	updatematrix
Update VSOE	updatevsoe
View All Transactions	viewalltransactions
Void	void
W4 Worksheet	w4data

Working with UI Objects

Applies to: SuiteScript 2.x | SuiteCloud Developer

You can use UI objects to build an assistant in NetSuite that has the same look-and-feel as other built-in NetSuite assistants.

For additional information about NetSuite Assistants, see Understanding NetSuite Assistants.

For Working with UI Objects in SuiteScript 2.0, see the help topic SuiteScript 2.x Working with UI Objects.

Understanding NetSuite Assistants

Applies to: SuiteScript 2.x | SuiteCloud Developer

In NetSuite, assistants contain a series of steps that users must complete to accomplish a larger task. In some assistants, users must complete the steps sequentially. In others, steps are non-sequential, and they do not all have to be completed. In these assistants, steps are provided only as guidelines for actions users might want to take to complete a larger task.

The UI objects you use to construct your own assistant will encapsulate the look-and-feel of assistants already built in to NetSuite. For examples of these assistants, see these topics:

- SuiteBundler Assistant
- Import Assistant

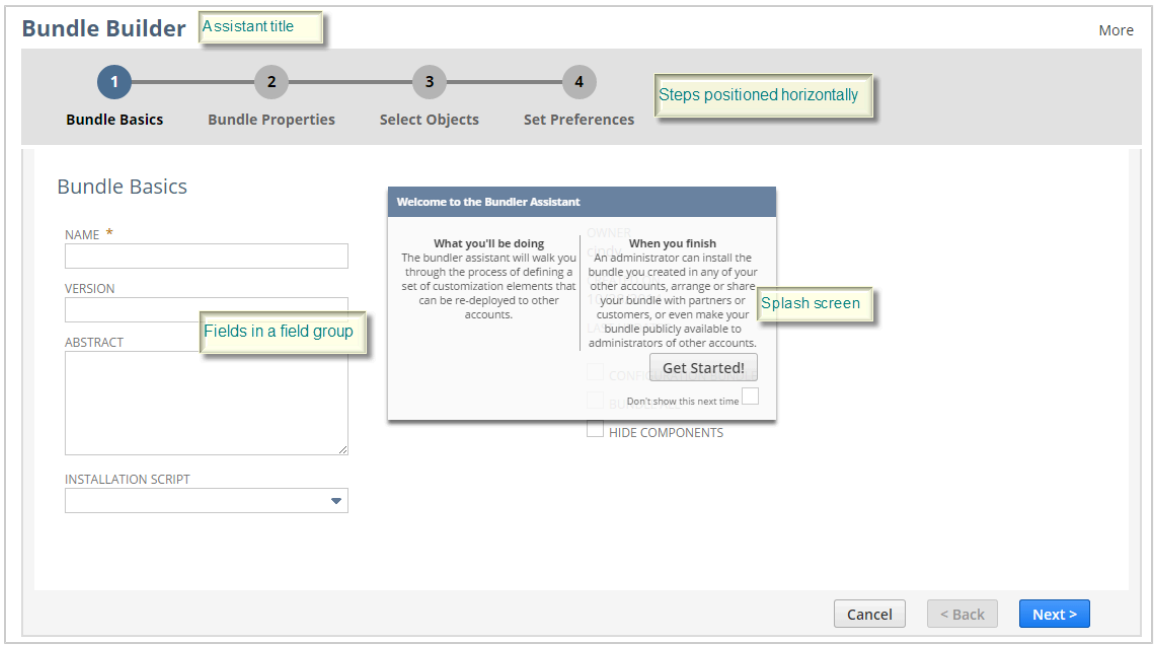
SuiteBundler Assistant

The SuiteBundler Assistant is another built-in NetSuite assistant. This assistant guides users through a set of steps in custom NetSuite solutions that are "bundled," later to be deployed into other NetSuite accounts.

Note: To access the SuiteBundler Assistant, go to Customization > SuiteBundler > Create Bundle.

This figure shows Step 1 (page 1) of the SuiteBundler Assistant. Steps are ordered sequentially and appear horizontally, directly below the title of the assistant.

All components called out in this figure can be built in a custom assistant.

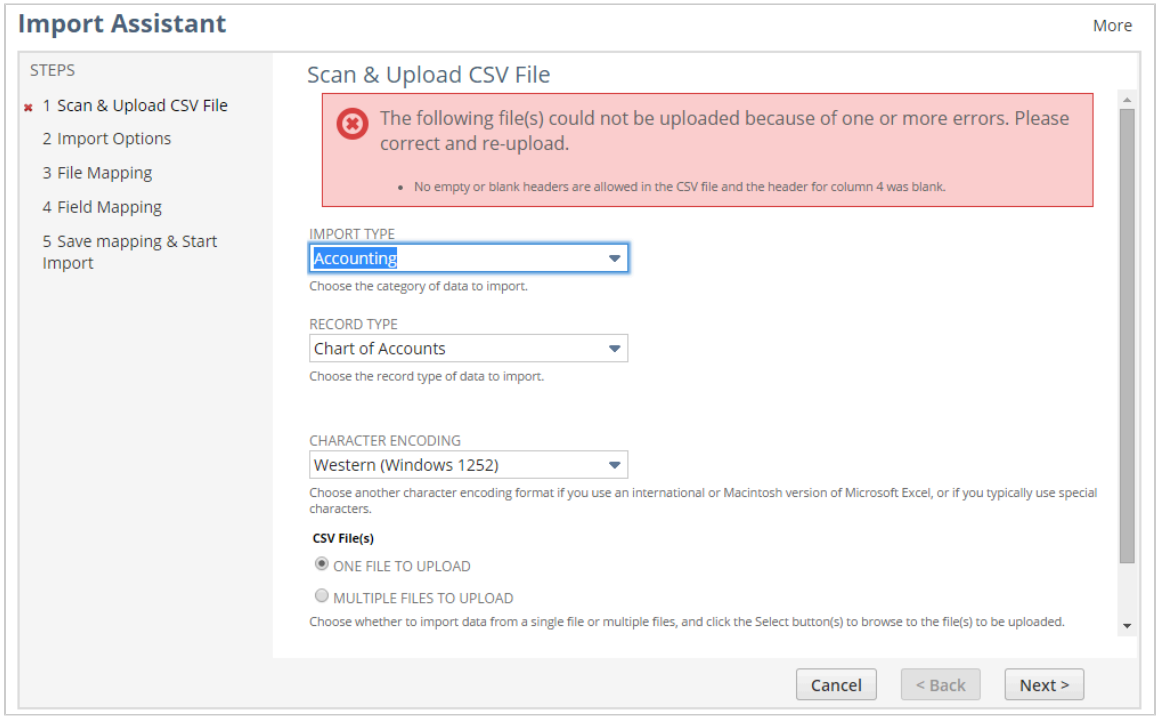


Import Assistant

The Import Assistant guides users through a set steps that allow them to import data into NetSuite.

Note: To access the Import Assistant, go to Setup > Import/Export > Import CSV Records.

The following figure shows what an error message looks like in an assistant. Users cannot proceed to the next step until the error is resolved. When building custom assistants, you can also throw errors that prevent users from moving to the next step.



Single Page Applications

Single Page Applications (SPAs) can be installed in your account as part of a SuiteApp installed from the Marketplace. The SPAs page displays a list of all of the SPAs in your account with the following information: its **Name**, a link to access the SPA URL, its **Description**, the name of the **SuiteApp** the SPA is part of, and the **SPA ID**.

From the list, you can access the management page where you can configure settings for the SPA. For more information about how to manage SPAs installed as part of a SuiteApp, see the help topic Managing Single Page Applications.

SuiteScript FAQ

Applies to: SuiteScript 2.x | APIs | SuiteScript 1.0 | SuiteCloud Developer

For FAQ information related to SOAP web services, see the help topic SOAP Web Services Frequently Asked Questions (FAQ).

General

How often is the SuiteScript API documentation updated?

The SuiteScript API documentation is frequently updated and pushed to the NetSuite Help Center.

For the most current API documentation, see the help topic SuiteScript 2.x API Reference.

Which records are supported in SuiteScript?

All information regarding SuiteScript-supported records, fields, tabs, sublists, search filters, and search columns is located in the NetSuite Help Center.

See the help topic SuiteScript Supported Records for information about supported records.

See the SuiteScript Records Browser for details related to SuiteScript-supported records.

Which sublists are supported in SuiteScript?

All information regarding SuiteScript-supported records, fields, tabs, sublists, search filters, and search columns is located in the SuiteScript Records Browser.

See the SuiteScript Records Browser for details related to SuiteScript-supported sublists and their internal IDs.

What is the difference between SuiteScript 1.0 and SuiteScript 2.x?

SuiteScript 1.0 uses global functions while SuiteScript 2.x is modular. The functions in SuiteScript 1.0 are synchronous on client, while SuiteScript 2.x can use asynchronous processing. SuiteScript 2.x offers more APIs. New functionality is being added to SuiteScript 2.x only.

See the SuiteScript 1.0 to SuiteScript 2.x API Map for more details on the differences.

A new version of SuiteScript, SuiteScript 2.1, is also available. This version is the latest minor version of SuiteScript. It extends SuiteScript 2.x by supporting additional ECMAScript language features and syntax. For more information, see the help topic SuiteScript 2.1.

What version of SuiteScript should I use?

Use SuiteScript 2.x for new scripts that you develop, and consider converting your SuiteScript 1.0 scripts to SuiteScript 2.0 or SuiteScript 2.1.

SuiteScript 1.0 is no longer being updated (no new feature development or enhancement work is being done). SuiteScript 1.0 scripts continue to be supported, but you should use SuiteScript 2.x for any new or substantially revised scripts to take advantage of new features, APIs, and functionality enhancements.

A new version of SuiteScript, SuiteScript 2.1, is also available. This version is the latest minor version of SuiteScript. It extends SuiteScript 2.x by supporting additional ECMAScript language features and syntax. For more information, see the help topic SuiteScript 2.1.

Does SuiteScript support the use of multiple library files?

Yes. Multiple library files can be loaded onto the Script record page. The Script record page is where you define the SuiteScript .js file you want to execute, as well as any associated library files that may be required when the SuiteScript code executes.

Also be aware that the system reads your library files in the order they appear on the Library Script File tab on the Script record page. For example, if your first library file references the second library file, an error will be thrown, since the first library file is loaded before the second.

If you do not know how to create a Script record, or where on the Script record you must load your library files, go to the topic Creating a Script Record in the NetSuite Help Center.

What is the difference between Standard and Dynamic mode?

When a SuiteScript 2.x script creates, copies, loads, or transforms a record in standard mode, the record's body fields and sublist line items are not sourced, calculated, and validated until the record is saved. In dynamic mode, the record's body fields and sublist line items are sourced, calculated, and validated in real-time.

See the help topic SuiteScript 2.x Standard and Dynamic Modes for more details.

What are the best practices when using Standard and Dynamic mode?

Standard mode is faster in many scenarios. It is better to use it if only read operation is needed.

Dynamic mode may be needed when doing write operations of fields on which other field depend.

See the help topic SuiteScript 2.x Standard and Dynamic Modes for more details.

How do I use require Configuration?

If you set up a valid `@NAmdConfig` JSDoc tag, SuiteScript implements the require configuration settings *before* loading dependencies.

See the help topic require Configuration for more details.

What is SuiteScript governance?

Governance serves like a sanity check of a script. If there is a bug in a script like unwanted infinite loop, governance prevents the script to run indefinitely.

To optimize application performance, NetSuite has implemented a SuiteScript governance model based on usage units. If the number of allowable units is exceeded, the script is terminated.

See the help topic SuiteScript Governance and Limits for more details.

What is SuiteCloud Processors?

SuiteCloud Processors is the current system used to process scheduled scripts and map/reduce scripts. Before SuiteCloud Processors was introduced, scheduled scripts and map/reduce scripts were exclusively processed by scheduling queues.

See SuiteCloud Processors for more details.

Which is better to use: Scheduled script or Map Reduce script?

Scheduled script is better for simple tasks which cannot run in parallel. Map/Reduce script is better suited for processing massive data in parallel, since it is faster then Scheduled script.

See the help topics SuiteScript 2.x Scheduled Script Type and SuiteScript 2.x Map/Reduce Script Type for more details.

What version of ECMAScript does SuiteScript support?

SuiteScript 1.0 and SuiteScript 2.0 support ECMAScript 5.1.

SuiteScript 2.1 supports ECMAScript 6. This version is the latest minor version of SuiteScript. It extends SuiteScript 2.x by supporting additional ECMAScript language features and syntax. For more information, see the help topic SuiteScript 2.1.

Can a User Event script be used to trigger another User Event script?

Nested user event script execution is not supported.

See the help topic SuiteScript 2.x User Event Script Type for more details.

SDF and SuiteApps

What is SDF and how does it help me develop with SuiteApps?

SuiteCloud Development Framework (SDF) is a development framework that you can use to create SuiteApps from an integrated development environment (IDE) on your local computer. Using SDF lets you manage client and server scripts as part of file-based customization projects. You can also use SDF with the SuiteCloud Software Developer Kit (SDK) as the best solution for creating a script as a part of an overall NetSuite customized solution packaged with other factors, such as custom records, custom forms, other scripts, or more. For more information about SDF, see the help topic SuiteCloud Development Framework,

Is there a special script type for installing SDF projects?

Yes. The SDF installation script type is used to perform tasks during development of a SuiteApp from SDF to your target account. You can use the SDF installation script type to perform setup, configuration, and data management tasks that otherwise would have to be completed by account administrators. For more information on the SDF installation script type, see the help topic SuiteScript 2.x SDF Installation Script Type.

Does SuiteScript have a module that I can use to see information about my SuiteApp?

Yes. You can use the N/suiteAppInfo module to access information related to your SuiteApp including whether a specific SuiteApp is installed and a lsit of all SDF SuiteApps that are installed, along with the ID for the SDF SuiteApp that contains a specific script (for multiple scripts specified). For more information on the N/suiteappInfo module, see the help topic N/suiteAppInfo Module.

Code

How can I control the execution order of scripts deployed on a record?

After logging in NetSuite, go to Customization > Scripting > Scripted Records. Select your record type. You can drag the scripts up or down. This step controls the order of their execution.

How many client and user event scripts can I deploy in a record?

There is no limit on the number of client or user event scripts that can be deployed. However, performance issues may occur if you have a lot of scripts deployed on a single record.

How do I add a button in a User Event script that redirects to a Suitelet in SuiteScript 2.x?

In the `beforeLoad` callback, one is supplied with a form object on which any button can be added. That button can execute an arbitrary function. The function must use `document.location = <url of a suitelet>` for redirect.

See the help topic SuiteScript 2.x User Event Script Type or SuiteScript 2.x Suitelet Script Type for more details.

Can I write to a script parameter field I have created in my code?

Although you can read the value of a script parameter field, you cannot programmatically set the value. The only time you can specify a value outside of the UI is when you call task.create(options).

```
var myTask = task.create({task.TaskType.SCHEDULED_SCRIPT});
myTask.scriptId = <id of your script>;
myTask.deploymentId = <id of your script deployment>;
myTask.params = <your params>;
myTask.submit();
```

See Creating Script Parameters Overview for more information about creating and working with script parameters.

How do I mass delete records using SuiteScript?

Currently, there is no SuiteScript function that enables you to mass delete records in one batch process. In SuiteScript, you have to delete each record individually using record.delete(options).

Which certificate authorities (CAs) does NetSuite support?

NetSuite supports the same list of trusted third-party certificate authorities (CAs) as Mozilla Included CA Certificate List.

The target endpoint, domain, or server must use one of these trusted third–party CAs, or the connection cannot be established. Oracle NetSuite requires that the endpoints that you are connecting to from NetSuite provide a full certification chain, including intermediate certificates.

For a list of certificate authorities, see https://wiki.mozilla.org/CA/Included_Certificates.

Can I throw an alert in a Suitelet?

You can attach a client script by using the N/ui/message Module.

```
/**
 * @NApiVersion 2.1
 * @NScriptType Suitelet
 */
define(['N/ui/serverWidget', 'N/ui/message'], function(serverWidget, message){
    function onRequest(context){
        let form = serverWidget.createForm('Alert Form');
        let msg = message.create({
            type:message.Type.WARNING,
            title:"Alert",
            message: "Something occurred",
            duration: 5000
        });
        form.addPageInitMessage(msg);
        context.response.writePage(form);
    }

```

```
    return{
        onRequest: onRequest
    }
});
```

N/ui/message Module and N/ui/dialog Module are executable in client scripts only.

Is there a change in creating subrecords in SuiteScript 2.x?

Yes. SuiteScript 2.x includes changes in how your script subrecords.

See the help topic Subrecord Scripting in SuiteScript 2.x Compared With 1.0 for more details.

How do I optimize SuiteScript performance?

There are several guidelines that you can follow to optimize SuiteScript performance.

See Optimizing SuiteScript Performance for more details.

Should I use record-level or form-level scripts?

There are different forms used for different users. They should be used when one needs to execute different scripts for different users. Record-level script is always used regardless of the user. Based on that, one can decide which type of script should be used.

See the help topic Record-Level and Form-Level Script Deployments for more details.

Where can I find the records and their scripts?

The Scripted Records page lists all the records that have user event or global client script that is associated with a record.

See the help topic The Scripted Records Page for more details.

What is the difference between Map Reduce and Scheduled Script?

The map/reduce script type is designed for scripts that need to handle large amounts of data. It is best suited for situations where the data can be divided into small, independent parts.

Scheduled scripts are server scripts that are executed (processed) with SuiteCloud Processors. It is run sequentially unlike Map/Reduce, where its stages are executed in parallel.

See the help topic SuiteScript 2.x Script Types for more details.

Is DOM supported in SuiteScript?

SuiteScript does not support direct access to the NetSuite UI through the Document Object Model (DOM). The NetSuite UI should only be accessed using SuiteScript APIs.

What are the required SuiteScript 2.x JSDoc Tags?

For entry point scripts, `NApiVersion` and `NScriptType` JSDoc tags are required. For more information about JSDoc tags, see the help topic: SuiteScript 2.x JSDoc Validation.

What are the stages required in a Map Reduce script?

At least `getInput` and one of map/reduce stages are required.

What does the script context seen in System Notes represent regarding the action on the record?

It is the context from which the operation was triggered that changed the field. So if you consider stack of operations, it is the second most nested scope.

For example, if in the UI, the `pageInit` callback is used on a record to call `nlapiRequestURL` to trigger a Suitelet. This calls another `nlapiRequestURL` to trigger a RESTlet, which loads a record that triggers a user event that finally manipulates the field, then the context is restlet. The stack will be `UI/Client/Suitelet/Restlet/UserEvent`. RESTlet is the second most nested scope.

How do you reference Custom Modules in SuiteScript 2.x?

You can add them by path to the dependencies of your `define` callback.

What is the difference between scriptContext.newRecord and scriptContext.oldRecord in SuiteScript 2.x?

`scriptContext.oldRecord` holds a copy of a record before any manipulation is completed on it. It is not available while creating new record.

`scriptContext.newRecord` contains the record which can be modified in the `beforeSubmit` callback to save the record differently than it was set in UI. For example, you can override a memo before the record is saved.

Additional tools to work with along with SuiteScript

What other tools are available to help me write my SuiteScript scripts?

Several tools are available, including SDF and SDK and the Records Catalog and the Records Browser. For more information on SDF and SDK, see the help topic SuiteCloud Development Framework. For more information about the Records Catalog and the Records Browser, see the help topics The Records Catalog and SuiteScript Records Browser. For more information about additional SuiteCloud tools, see the help topic SuiteCloud Reference Tools.

Errors

You can find SuiteScript error descriptions in SuiteAnswers. Type the error in the search box and click **Search**.

How does “Record Has Been Changed” error get triggered in SuiteScript execution?

This error can happen when another user or script has manipulated the record in parallel.

See the SuiteAnswers article Prevent losing data because of the error "Record has been changed" for more details.

What is the “You do not have privileges to view this page” error message on the script deployments page?

End users accessing Suitelets will receive this error if any of the **Select All** box on the Audience tab on the Script Deployment page was not selected. This is true even for users who are accessing the Suitelet externally (for example, users who are accessing a Suitelet using the “external” Available Without Login URL for the Suitelet.)

See Errors Related to the Available Without Login URL for more details.

What is the "You are not allowed to navigate directly to this page" error message when accessing Suitelet?

End users accessing Suitelets through the Available Without Log external URL will receive this error if the **Status** field on the Suitelet's Script Deployment page has not been set to Released.

See Errors Related to the Available Without Login URL for more details.

Why am I getting an "SSS_USAGE_LIMIT_EXCEEDED ID" error?

The script exceeded its usage point limit for a single execution. If this error occurs twice, the script execution will stop.

For more information about usage limits, see the help topic SuiteScript Governance and Limits

It is best that you monitor your script usage because there are limits. For more information, see Monitoring Script Usage.

Why am I getting an "SSS_REQUEST_LIMIT_EXCEEDED" error?

This error is thrown if a single execution of a server script or application has taken longer than its limit.

For more information about usage limits, see the help topic SuiteScript Governance and Limits

Why am I getting "SSS_INVALID_SUBLIST_OPERATION" error?

A line item field value is being set on a non-existing line. The sublist doesn't exist or is not editable.

See the SuiteAnswers article SuiteScript Error SSS_INVALID_SUBLIST_OPERATION for more details.

Logs

How long will the script execution logs be available on my account?

There are two locations to find script execution logs:

- **Execution Log** tab on Script and Script Deployment records — The capacity for script execution logs on the Execution Log tab is shared by customers on the same database. For protection against excessive logging, script execution logs are governed by a total storage limit on each instance of the NetSuite database. On each NetSuite server, if the database table that stores logs reaches this limit, all logs (across all customers on that server) are purged. This means that you may have logs past 30 days if the volume is low, but you may have logs of less than a week if the log volume is extremely large. For this reason, you should store information from logs that you need to save using custom records.
- **Script Execution Log** page — The execution logs at Customization > Scripting > Script Execution Logs store logs for up to 30 days, regardless of volume. However, there is a limit of 10,000 results that can be displayed on this page. If you have more than 10,000 logs in 30 days, you can view these logs by filtering the Script Execution Log page. Click the **Filters** section at the top of the page to expand the filters and limit to date range, script, or log level to access the logs you need. Note that a full search or Customize View option is not available for this page.

Can I use execution logs in a client script?

Yes. For SuiteScript 1.0, use `nlapiLogExecution` (see the help topic SuiteScript 1.0 Documentation). For SuiteScript 2.x, use N/log Module.