should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described in this document 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.

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc

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

Oracle may make available to Customer certain features that are labeled “beta” that are not yet generally available. To use such features, Customer acknowledges and agrees that such beta features are subject to the terms and conditions accepted by Customer upon activation of the feature, or in the absence of such terms, subject to the limitations for the feature described in the User Guide and as follows: The beta feature is a prototype or beta version only and is not error or bug free and Customer agrees that it will use the beta feature carefully and will not use it in any way which might result in any loss, corruption or unauthorized access of or to its or any third party's property or information. Customer must promptly report to Oracle any defects, errors or other problems in beta features to support@netsuite.com or other designated contact for the specific beta feature. Oracle cannot guarantee the continued availability of such beta features and may substantially modify or cease providing such beta features without entitling Customer to any refund, credit, or other compensation. Oracle makes no representations or warranties regarding functionality or use of beta features and Oracle shall have no liability for any lost data, incomplete data, re-run time, inaccurate input, work delay, lost profits or adverse effect on the performance of the Service resulting from the use of beta features. Oracle's standard service levels, warranties and related commitments regarding the Service shall not apply to beta features and they may not be fully supported by Oracle's customer support. These limitations and exclusions shall apply until the date that Oracle at its sole option makes a beta feature generally available to its customers and partners as part of the Service without a “beta” label.
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

How to Use SuiteScript Records Help ................................................................. 1
SuiteScript Supported Records ........................................................................ 2

### Activities
- Activity ........................................................................................................ 12
- Event ........................................................................................................... 12
- Phone Call ................................................................................................. 13
- Project Task .............................................................................................. 14
- Resource Allocation .................................................................................. 18
- Task ........................................................................................................... 20
- Work Calendar .......................................................................................... 20

### Entities
- Bonus .......................................................................................................... 21
- Bonus Type ............................................................................................... 22
- Competitor ............................................................................................... 25
- Contact ....................................................................................................... 26
- Customer ................................................................................................... 27
- Customer Status ....................................................................................... 27
- Employee ................................................................................................. 30
- Employee Status ..................................................................................... 30
- Employee Type ....................................................................................... 32
- Entity .......................................................................................................... 33
- Generic Resource ..................................................................................... 34
- Job ............................................................................................................. 34
- Lead ........................................................................................................... 35
- Other Name ............................................................................................. 37
- Partner ...................................................................................................... 38
- Project ....................................................................................................... 40
- Project Status .......................................................................................... 41
- Project Template ...................................................................................... 42
- Project Type ............................................................................................. 43
- Prospect .................................................................................................... 44
- Time-Off Management ........................................................................... 46
- Vendor ....................................................................................................... 47

### Items
- Using Item Records in SuiteScript .............................................................. 56
- Pricing Sublist / Pricing Matrix ................................................................ 57
- Assembly Item .......................................................................................... 58
- Description Item ....................................................................................... 59
- Discount Item ........................................................................................... 60
- Download Item ........................................................................................ 61
- Gift Certificate Item ................................................................................ 62
- Inventory Item .......................................................................................... 63
- Item Group ............................................................................................... 64
- Item Search ............................................................................................... 65
- Kit Item ..................................................................................................... 66
- Lot Numbered Assembly Item .................................................................. 67
- Lot Numbered Inventory Item .................................................................. 68
- Markup Item ............................................................................................. 69
- Non-Inventory Part .................................................................................. 70
- Other Charge Item .................................................................................. 71
- Payment Item ........................................................................................... 72
- Reallocate Items ...................................................................................... 73
- Serialized Assembly Item ....................................................................... 74

---

### Additional Sections

- Activities
- Entities
- Items
Lists .............................................................................................................................. 252
Support .........................................................................................................................  246
Billing Rate Card   .........................................................................................................  266
Billing Class  ...............................................................................................................  265
Billing Account  ...........................................................................................................  264
Amortization Template  ................................................................................................. 262
Amortization Schedule  .................................................................................................  261
Allocation Schedule  .....................................................................................................  259
Accounting Period  ......................................................................................................  258
Accounting Context  .....................................................................................................  257
Account  .....................................................................................................................  254
Folder  .......................................................................................................................  251
Topic  ........................................................................................................................  250
Issue  ........................................................................................................................  247
Case ...........................................................................................................................  246
Work Order Issue   .......................................................................................................  244
Work Order Close   .......................................................................................................  242
Work Order Completion .................................................................................................  243
Work Order  ...............................................................................................................  240
Support .........................................................................................................................  246
Statistical Journal Entry   ..............................................................................................  228
Vendor Prepayment Application   ....................................................................................  227
Vendor Prepayment  ....................................................................................................  225
Vendor Credit  ............................................................................................................  221
Vendor Bill ....................................................................................................................  219
Vendor  .........................................................................................................................  219
Usage  .......................................................................................................................  219
Unlocked Time Period   .................................................................................................  217
Transfer Order  ...........................................................................................................  216
Transfer Order Search .................................................................................................  215
Transaction Search ......................................................................................................  214
Transaction Actions .....................................................................................................  213
Time ...............................................................................................................................  213
Subscription Line  ........................................................................................................  212
Subscription ..................................................................................................................  212
Sales Order ...................................................................................................................  204
Revenue Commitment Reversal .......................................................................................  203
Revenue Commitment .................................................................................................  203
Revenue Arrangement .................................................................................................  200
Revenue Order ............................................................................................................  200
Revenue ....................................................................................................................  200
Period End Journal ........................................................................................................  193
Rental Card ....................................................................................................................  193
Return Authorization .....................................................................................................  198
Requisition ..................................................................................................................  196
Return Authorization .....................................................................................................  198
Purchase Order ............................................................................................................  194
Purchase Contract ........................................................................................................  194
Purchase Order ............................................................................................................  194
Purchase Contract ........................................................................................................  194
Purchase ........................................................................................................................  194
Paycheck Journal ..........................................................................................................  186
Paycheck ........................................................................................................................  185
Payroll Batch ................................................................................................................  189
Payroll Batch Employee ...............................................................................................  191
Price Plan .....................................................................................................................  192
Period End Journal ........................................................................................................  193
Order Schedule ...........................................................................................................  184
Paycheck ........................................................................................................................  185
Paycheck Journal ..........................................................................................................  186
Payroll Batch ................................................................................................................  189
Payroll Batch Employee ...............................................................................................  191
Price Plan .....................................................................................................................  192
Period End Journal ........................................................................................................  193
Purchase Contract ........................................................................................................  194
Purchase Order ............................................................................................................  194
Purchase ........................................................................................................................  194
Payment Plan ................................................................................................................  193
Payroll ...........................................................................................................................  192
Order ..............................................................................................................................  191
Order ..............................................................................................................................  191
Order Search ..................................................................................................................  190
Order Plan ......................................................................................................................  189
Order ..............................................................................................................................  189
Order Search ..................................................................................................................  189
Order Plan ......................................................................................................................  188
Order ..............................................................................................................................  188
Order Search ..................................................................................................................  188
Order Plan ......................................................................................................................  187
Order ..............................................................................................................................  187
Order Search ..................................................................................................................  187
Order Plan ......................................................................................................................  186
Order ..............................................................................................................................  186
Order Search ..................................................................................................................  186
Order Plan ......................................................................................................................  185
Order ..............................................................................................................................  185
Order Search ..................................................................................................................  185
Order Plan ......................................................................................................................  184
How to Use SuiteScript Records Help

**Note:** This topic applies to all versions of SuiteScript.

The SuiteScript Records Guide must be used in conjunction with the SuiteScript Records Browser. Each record listed in this guide includes a link to the Records Browser. Within the Records Browser, you will find all available fields, sublists, and search filter fields for that record. The Records Browser also provides field level help for many fields that appears on the record. Within this guide, the description of each record often includes the following types of information:

- **Supported script types:** A description of whether the record is supported in both client and server SuiteScript, and whether the user events are supported
- **Supported functions:** A list of all functions that can be used with the record
- **Field definitions:** A link to the Records Browser and, in some cases, additional explanations about how to work with particular fields
- **Usage notes:** Additional details or context about working with the record

**Important:** For a list of all NetSuite records that are supported in SuiteScript, see [SuiteScript Supported Records](#).

**Working Online**

If you are working online, using either the PDF or online help version of this guide, click the links to the SuiteScript Records Browser that are provided with each record description. The Records Browser will open in your default browser.

**Downloading the SuiteScript Records Browser**

If you want to use the SuiteScript Records Browser when working offline, you must download the [SuiteScript Records Browser .zip file](#).

After downloading the .zip file to your local machine, extract the file contents and navigate to the script directory. To view the Records Browser content, open index.html file in the browser of your choice.

For information on using the Records Browser, see the help topic [Working with the SuiteScript Records Browser](#).
## SuiteScript Supported Records

### Note:
The topic applies to all versions of SuiteScript.

The following table lists all NetSuite records that are scriptable with SuiteScript. This table includes:

- clickable record names, which take you to detailed information about the record
- record IDs, which are often referenced in SuiteScript APIs
- scripting levels (where Full scripting level means that the record can be created, updated, copied, deleted, and searched)
- SuiteScript client and/or server support

Also note that all subrecords are scriptable from the line item level, unless stated otherwise.

The following help topics contain additional information on using the SuiteScript Records Browser:

- Working with the SuiteScript Records Browser
- How to Use SuiteScript Records Help

<table>
<thead>
<tr>
<th>Record Name</th>
<th>Record ID</th>
<th>Record Category</th>
<th>Scripting Level</th>
<th>Scriptable in Client Suite Script</th>
<th>Scriptable in Server SuiteScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Account</td>
<td>account</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Accounting Book</td>
<td>accountingbook</td>
<td>List</td>
<td>Copy Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Accounting Context</td>
<td>accountingcontext</td>
<td>List</td>
<td>Read, Create, Update, Copy, Delete, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Accounting Period</td>
<td>accountingperiod</td>
<td>List</td>
<td>Read and Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Activity</td>
<td>activity</td>
<td>Activity</td>
<td>Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Address</td>
<td>addressbookaddress</td>
<td>Subrecord</td>
<td>See the help topic Using SuiteScript with Address Subrecords</td>
<td>X</td>
<td>Server-side scripts must access through the parent record.</td>
</tr>
<tr>
<td>Advanced Intercompany Journal Entry</td>
<td>advintercompanyjournalentry</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Allocation Schedule</td>
<td>allocationschedule</td>
<td>List</td>
<td>Create, Read, Update, and Delete</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Amortization Schedule</td>
<td>amortizationschedule</td>
<td>List</td>
<td>Copy, Create, and Delete Not Allowed</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Amortization Template</td>
<td>amortizationtemplate</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Assembly Build</td>
<td>assemblybuild</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Assembly Item</td>
<td>assemblyitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Assembly Unbuild</td>
<td>assemblyunbuild</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Balance Transactions by Segments</td>
<td>balancetransbysegments</td>
<td>Transaction</td>
<td>Create, Read, and Search</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Billing Account</td>
<td>billingaccount</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Billing Class</td>
<td>billingclass</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Billing Rate Card</td>
<td>billingratecard</td>
<td>List</td>
<td>Read, Create, Update, Delete, and Search</td>
<td>X</td>
<td>Copy and Transform Not Supported</td>
</tr>
<tr>
<td>Billing Schedule</td>
<td>billingschedule</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client SuiteScript</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Bin</td>
<td>bin</td>
<td>List</td>
<td>Full</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Bin Putaway Worksheet</td>
<td>binworksheet</td>
<td>Transaction</td>
<td>Copy and Update Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Bin Transfer</td>
<td>bintransfer</td>
<td>Transaction</td>
<td>Full</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Blanket Purchase Order</td>
<td>blanketpurchaseorder</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>BOM</td>
<td>bom</td>
<td>List</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>BOM Revision</td>
<td>bomrevision</td>
<td>List</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bonus</td>
<td>bonus</td>
<td>Entity</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Bonus Type</td>
<td>bonustype</td>
<td>Entity</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Budget Exchange Rate</td>
<td>budgetexchangerate</td>
<td>List</td>
<td>Read, Update, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Campaign</td>
<td>campaign</td>
<td>Marketing</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Campaign Response</td>
<td>campaignresponse</td>
<td>Marketing</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Campaign Template</td>
<td>campaigntemplate</td>
<td>Marketing</td>
<td>Search Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Case</td>
<td>supportcase</td>
<td>Support</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cash Refund</td>
<td>cashrefund</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cash Sale</td>
<td>cashsale</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Change Order</td>
<td>subscriptionchangeorder</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Charge</td>
<td>charge</td>
<td>Transaction</td>
<td>Full</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Check</td>
<td>check</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Class</td>
<td>classification</td>
<td>List</td>
<td>Server SuiteScript Only</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CMS Content</td>
<td>cmscontent</td>
<td>Web Site</td>
<td>Create, Update, and Search Only Attach, Transform, and Copy Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CMS Content Type</td>
<td>cmscontenttype</td>
<td>Web Site</td>
<td>Create, Update, and Search Only Attach, Transform, and Copy Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>CMS Page</td>
<td>cmspage</td>
<td>Web Site</td>
<td>Create, Update, and Search Only Attach, Transform, and Copy Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Commerce Category</td>
<td>commercecategory</td>
<td>Website</td>
<td>Attach, Transform, and Copy Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Competitor</td>
<td>competitor</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client Suite Script</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------------------------</td>
<td>-----------------</td>
<td>----------------------------------------</td>
<td>-----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Consolidated Exchange Rate</td>
<td>consolidatedexchangerate</td>
<td>List</td>
<td>Read, Update, and Search Only Create and Delete Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Contact</td>
<td>contact</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Contact Category</td>
<td>contactcategory</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Contact Role</td>
<td>contactrole</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Cost Category</td>
<td>costcategory</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Coupon Code</td>
<td>couponcode</td>
<td>Marketing</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Credit Card Charge</td>
<td>creditcardcharge</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Credit Card Refund</td>
<td>creditcardrefund</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Credit Memo</td>
<td>creditmemo</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Currency</td>
<td>currency</td>
<td>List</td>
<td>Full — with Multiple Currencies Feature Enabled Read Only — without Multiple Currencies Feature</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Custom List</td>
<td>See Custom List IDs</td>
<td>Custom</td>
<td>Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Custom Record</td>
<td>See Custom Record IDs</td>
<td>Custom</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer</td>
<td>customer</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Category</td>
<td>customercategory</td>
<td>List</td>
<td>Search Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Deposit</td>
<td>customerdeposit</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Message</td>
<td>customermessage</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer Payment</td>
<td>customerpayment</td>
<td>Transaction</td>
<td>Copy and Create Not Allowed</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer Refund</td>
<td>customerrefund</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer Status</td>
<td>customerstatus</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Customer-Subsidiary Relationship</td>
<td>customersubsidiary</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Department</td>
<td>department</td>
<td>List</td>
<td>Server SuiteScript Only</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Deposit</td>
<td>deposit</td>
<td>Transaction</td>
<td>Copy Not Supported Dynamic Mode Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Deposit Application</td>
<td>depositapplication</td>
<td>Transaction</td>
<td>Create Not Allowed</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Description Item</td>
<td>descriptionitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Discount Item</td>
<td>discountitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Download Item</td>
<td>downloaditem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Driver's License</td>
<td>driverslicense</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Email Template</td>
<td>emailtemplate</td>
<td>Marketing</td>
<td>Search Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Employee</td>
<td>employee</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client Suite Script</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>------------------------------------</td>
<td>-----------------</td>
<td>----------------</td>
<td>-----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Employee Change Request</td>
<td>employeechangerequest</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Employee Change Request Type</td>
<td>employeechangerequesttype</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Employee Status</td>
<td>employeestatus</td>
<td>Entity</td>
<td>Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Employee Type</td>
<td>employeetype</td>
<td>Entity</td>
<td>Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Entity</td>
<td>entity</td>
<td>Entity</td>
<td>Search Only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estimate</td>
<td>estimate</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Event</td>
<td>calendarevent</td>
<td>Activity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Expense Category</td>
<td>expensecategory</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Expense Report</td>
<td>expensereport</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fair Value Price</td>
<td>fairvaluepricelist</td>
<td>List</td>
<td>Create, Delete, and Search Only Copy Not Allowed</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>File</td>
<td>file</td>
<td>File Cabinet</td>
<td>Search Only</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Financial Institution</td>
<td>financialinstitution</td>
<td>List</td>
<td>Read, Create, Update, Delete, and Search</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Folder</td>
<td>folder</td>
<td>File Cabinet</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Format Profile</td>
<td>formatprofile</td>
<td>List</td>
<td>Read, Create, Update, and Delete</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Fulfillment Request</td>
<td>fulfillmentrequest</td>
<td>Transaction</td>
<td>Create and Copy Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>General Token</td>
<td>generaltoken</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Gift Certificate Item</td>
<td>giftcertificateitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>GL Audit Numbering Sequence</td>
<td>glnumberingsequence</td>
<td>Transaction</td>
<td>Read, Create, Update, Copy, Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Global Account Mapping</td>
<td>globalaccountmapping</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Global Inventory Relationship</td>
<td>globalinventoryrelationship</td>
<td>List</td>
<td>Read, Create, Edit, Delete, Search Copy Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Group</td>
<td>entitygroup</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Inbound Shipment</td>
<td>inboundshipment</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Intercompany Allocation Schedule</td>
<td>intercompallocationschedule</td>
<td>List</td>
<td>Create, Read, Update, and Delete</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Intercompany Journal Entry</td>
<td>intercompanyjournalentry</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Intercompany Transfer Order</td>
<td>intercompanytransferorder</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Inventory Adjustment</td>
<td>inventoryadjustment</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
## SuiteScript Supported Records

<table>
<thead>
<tr>
<th>Record Name</th>
<th>Record ID</th>
<th>Record Category</th>
<th>Scripting Level</th>
<th>Scriptable in Client Suite Script</th>
<th>Scriptable in Server SuiteScript</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inventory Cost Revaluation</td>
<td>inventorycostrevaluation</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Inventory Count</td>
<td>inventorycount</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Inventory Detail</td>
<td>inventorydetail</td>
<td>Subrecord</td>
<td>See the help topic Scripting the Inventory Detail Subrecord</td>
<td>X</td>
<td>Server-side scripts must access through the parent record</td>
</tr>
<tr>
<td>Inventory Item</td>
<td>inventoryitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>(Also referred to in the UI as Inventory Part)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Inventory Number</td>
<td>inventorynumber</td>
<td>List</td>
<td>Copy, Create, and Delete Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Inventory Status</td>
<td>inventorystatus</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Inventory Status Change</td>
<td>inventorystatuschange</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Inventory Transfer</td>
<td>inventorytransfer</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Invoice</td>
<td>invoice</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Issue</td>
<td>issue</td>
<td>Support</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Issue Product</td>
<td>issueproduct</td>
<td>Support</td>
<td>Copy, transform and search not supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Issue Product Version</td>
<td>productversion</td>
<td>Support</td>
<td>Copy, transform and search not supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Account Mapping</td>
<td>itemaccountmapping</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Collection</td>
<td>itemcollection</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Collection Item Map</td>
<td>itemcollectionmap</td>
<td>List</td>
<td>Update, Copy, and Transform Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Demand Plan</td>
<td>itemdemandplan</td>
<td>Transaction</td>
<td>Copy Not Supported, No Available Transforms</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Fulfillment</td>
<td>itemfulfillment</td>
<td>Transaction</td>
<td>Copy and Create Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Group</td>
<td>itemgroup</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Location Configuration</td>
<td>itemlocationconfiguration</td>
<td>List</td>
<td>Read, Create, Edit, Delete, and Search</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Receipt</td>
<td>itemreceipt</td>
<td>Transaction</td>
<td>Copy and Create Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Revision</td>
<td>itemrevision</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Item Search</td>
<td>item</td>
<td>Item</td>
<td>Search Only</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Item Supply Plan</td>
<td>itemsupplyplan</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Job</td>
<td>job</td>
<td>Entity</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Job Requisition</td>
<td>jobrequisition</td>
<td>Entity</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Journal Entry</td>
<td>journalentry</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Kit Item</td>
<td>kititem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Kudos Feature Records</td>
<td>kudos</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client SuiteScript</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Landed Cost</td>
<td>landedcost</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Lead</td>
<td>lead</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Location</td>
<td>location</td>
<td>List</td>
<td>Server SuiteScript Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Lot Numbered Assembly Item</td>
<td>lotnumberedassem blyitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Lot Numbered Inventory Item</td>
<td>lotnumberedinventoryitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Manufacturing Cost Template</td>
<td>manufacturingcost template</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Manufacturing Operation Task</td>
<td>manufacturingoper ationtask</td>
<td>Transaction</td>
<td>Copy Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Manufacturing Planned Time</td>
<td>mfgplannedtime</td>
<td>Transaction</td>
<td>Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Manufacturing Routing</td>
<td>manufacturingrout ing</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Markup Item</td>
<td>markupitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Memorized Transaction Definition</td>
<td>memdoc</td>
<td>Transaction</td>
<td>Read, Update, Delete, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Merchandise Hierarchy Level</td>
<td>merchandisehiera rylevel</td>
<td>List</td>
<td>Read, Create, Edit, Delete, and Search</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Merchandise Hierarchy Node</td>
<td>merchandisehiera rychynode</td>
<td>List</td>
<td>Read, Create, Edit, Delete, and Search</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Merchandise Hierarchy Version</td>
<td>merchandisehiera rychyversion</td>
<td>List</td>
<td>Read, Create, Edit, Delete, and Search</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Message</td>
<td>message</td>
<td>Communication</td>
<td>For details, see Message.</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Multi-Book Accounting Transaction</td>
<td>accountingtransac tion</td>
<td>Transaction</td>
<td>Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Nexus</td>
<td>nexus</td>
<td>List</td>
<td>Copy Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Non-Inventory Part</td>
<td>noninventoryitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Note</td>
<td>note</td>
<td>Communication</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Note Type</td>
<td>notetype</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Opportunity</td>
<td>opportunity</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Order Schedule</td>
<td>orderschedule</td>
<td>Transaction</td>
<td>See the help topic Working with Subrecords in SuiteScript.</td>
<td>X</td>
<td>Server-side scripts must access through the parent record</td>
</tr>
<tr>
<td>Organization Value</td>
<td>organizationva lue</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Other Charge Item</td>
<td>otherchargeitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Other Government-Issued ID</td>
<td>othergovernmentissuedid</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>User event scripts are not supported</td>
</tr>
<tr>
<td>Other Name</td>
<td>othername</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Other Name Category</td>
<td>othernamecategory</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Ownership Transfer</td>
<td>bulkownershiptransfer</td>
<td>List</td>
<td>Create, Delete, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client Suite Script</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>-------------------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>OwnershipTransfer</td>
<td>partner</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Partner Category</td>
<td>partnercategory</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Passport</td>
<td>passport</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Paycheck</td>
<td>paycheck</td>
<td>Transaction</td>
<td>Read and Update Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Paycheck Journal</td>
<td>paycheckjournal</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Payment Card</td>
<td>paymentcard</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Payment Card Token</td>
<td>paymentcardtoken</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Payment Item</td>
<td>paymentitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Payment Instruments</td>
<td>paymentcard, paymentcardtoken, generaltoken</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Payment Method</td>
<td>paymentmethod</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Payroll Batch</td>
<td>payroll batch</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Payroll Batch Employee</td>
<td>payrollbatchaddemployees</td>
<td>Transaction</td>
<td>Read and Update Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Payroll Item</td>
<td>payrollitem</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Period End Journal</td>
<td>periodendjournal</td>
<td>Transaction</td>
<td>Read, Search, limited Update</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Phone Call</td>
<td>phonecall</td>
<td>Activity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Price Book</td>
<td>pricebook</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Price Level</td>
<td>pricelevel</td>
<td>List</td>
<td>Search Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Price Plan</td>
<td>priceplan</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Pricing Group</td>
<td>pricinggroup</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Project</td>
<td>job</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Project Charge Rule</td>
<td>chargerule</td>
<td>List</td>
<td>Transform Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Project Expense Type</td>
<td>projectexpense</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Project Revenue Rule</td>
<td>laborbasedprojectrevenuerule</td>
<td>List</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Project Status</td>
<td>jobstatus</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Project Task</td>
<td>projecttask</td>
<td>Event</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Project Template</td>
<td>projecttemplate</td>
<td>Entity</td>
<td>Copy Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Project Type</td>
<td>jobtype</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Project Work Breakdown Structure</td>
<td>wbs</td>
<td>List</td>
<td>Copy and Transform not supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Promotion</td>
<td>promotioncode</td>
<td>Marketing</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client Suite Script</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------</td>
<td>-----------------</td>
<td>-----------------</td>
<td>-----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Prospect</td>
<td>prospect</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Purchase Contract</td>
<td>purchasecontract</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>purchaseorder</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Reallocate Items</td>
<td>reallocateItem</td>
<td>Item</td>
<td>Exposed for user event scripts only.</td>
<td>This record can not be created, loaded, or deleted using scripts.</td>
<td></td>
</tr>
<tr>
<td>Requisition</td>
<td>purchasersrequisition</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Resource Allocation</td>
<td>resourceallocation</td>
<td>Transactions</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Return Authorization</td>
<td>returnauthorization</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Revenue Arrangement</td>
<td>revenuearrangement</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Revenue Commitment</td>
<td>revenuecommitment</td>
<td>Transaction</td>
<td>Create Not Supported</td>
<td>See Usage Notes.</td>
<td>X</td>
</tr>
<tr>
<td>Revenue Commitment Reversal</td>
<td>revenuecommitmentreversal</td>
<td>Transaction</td>
<td>Create Not Supported</td>
<td>See Usage Notes.</td>
<td>X</td>
</tr>
<tr>
<td>Revenue Recognition Event</td>
<td>billingrevenueevent</td>
<td>List</td>
<td>Create, Red, Update, Delete, and Search</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Revenue Recognition Plan</td>
<td>revrecplan</td>
<td>List</td>
<td>Copy, Create, and Delete Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Revenue Recognition Schedule</td>
<td>revrecschedule</td>
<td>List</td>
<td>Copy, Create, and Delete Not Supported</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Revenue Recognition Template</td>
<td>revrectemplate</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Role</td>
<td>role</td>
<td>List</td>
<td>Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sales Order</td>
<td>salesorder</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sales Role</td>
<td>salesrole</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Sales Tax Item</td>
<td>salestaxitem</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Scheduled Script Instance</td>
<td></td>
<td>Customization</td>
<td>Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Script</td>
<td></td>
<td>Need Internal ID. See Script description</td>
<td>Customization</td>
<td>Copy, Create, and Delete Not Supported</td>
<td>See Usage Notes</td>
</tr>
<tr>
<td>Script Deployment</td>
<td></td>
<td>Need Internal ID. See Script Deployment description</td>
<td>Customization</td>
<td>See Usage Notes</td>
<td>X</td>
</tr>
<tr>
<td>Serialized Assembly Item</td>
<td>serializedassemblyitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Serialized Inventory Item</td>
<td>serializedinventoryitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Service</td>
<td>serviceitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client SuiteScript</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------</td>
<td>-----------------</td>
<td>----------------------------------</td>
<td>-----------------------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Shipping Item</td>
<td>shipitem</td>
<td>Item</td>
<td>Copy and Create Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Shopping Cart</td>
<td>shoppingcart</td>
<td>Website</td>
<td>Search Only</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Solution</td>
<td>solution</td>
<td>Support</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Statistical Journal Entry</td>
<td>statisticaljournalentry</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Store Pick Up Fulfillment</td>
<td>storepickupfulfillment</td>
<td>Transaction</td>
<td>Copy and Create Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subscription</td>
<td>subscription</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subscription Line</td>
<td>subscriptionline</td>
<td>Transaction</td>
<td>Create and Delete Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subscription Plan</td>
<td>subscriptionplan</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subsidiary</td>
<td>subsidiary</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Subsidiary Settings</td>
<td>subsidiarysettings</td>
<td>List</td>
<td>Read, Update, and Search</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Subtotal Item</td>
<td>subtotalitem</td>
<td>Item</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Supply Chain Snapshot</td>
<td>supplychainsnapshot</td>
<td>List</td>
<td>Read, Create, Edit, Delete, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Supply Chain Snapshot Simulation</td>
<td>supplychainsnapshot</td>
<td>Item</td>
<td>Read, Create, Update, Delete, Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>System Note</td>
<td>systemnote</td>
<td>(SYSTEM_NOTE for SuiteScript 2.0 search enum)</td>
<td>List</td>
<td>Search Only</td>
<td>X</td>
</tr>
<tr>
<td>Task</td>
<td>task</td>
<td>Activity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tax Control Account</td>
<td>taxacct</td>
<td>List</td>
<td>Delete and Search Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Tax Group</td>
<td>taxgroup</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Tax Period</td>
<td>taxperiod</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Tax Type</td>
<td>taxtype</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td>term</td>
<td>List</td>
<td>Search Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Time</td>
<td>timebill</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Time-Off Change</td>
<td>timeoffchange</td>
<td>Entity</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>User Event Triggering not supported.</td>
</tr>
<tr>
<td>Time-Off Plan</td>
<td>timeoffplan</td>
<td>Entity</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>User Event Triggering not supported.</td>
</tr>
<tr>
<td>Time-Off Request</td>
<td>timeoffrequest</td>
<td>Entity</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>User Event Triggering not supported.</td>
</tr>
<tr>
<td>Time-Off Rule</td>
<td>timeoffrule</td>
<td>Entity</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Record Name</td>
<td>Record ID</td>
<td>Record Category</td>
<td>Scripting Level</td>
<td>Scriptable in Client Suite</td>
<td>Scriptable in Server SuiteScript</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-----------------------</td>
<td>----------------------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>Time-Off Type</td>
<td>timeofftype</td>
<td>Entity</td>
<td>Copy and Transform Not Supported</td>
<td>X</td>
<td>User Event Triggering not supported.</td>
</tr>
<tr>
<td>Topic</td>
<td>topic</td>
<td>Support</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Transaction Search</td>
<td>transaction</td>
<td>Transaction</td>
<td>Search Only</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Transfer Order</td>
<td>transferorder</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Unit of Measure</td>
<td>unitstype</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Unlocked Time Period</td>
<td>unlockedtimeperiod</td>
<td>Transaction</td>
<td>Read, Create, Update, Delete, Copy, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Usage</td>
<td>usage</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor</td>
<td>vendor</td>
<td>Entity</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Bill</td>
<td>vendorbill</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Category</td>
<td>vendorcategory</td>
<td>List</td>
<td>Search Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Credit</td>
<td>vendorcredit</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Payment</td>
<td>vendorpayment</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Prepayment</td>
<td>vendoreprepayment</td>
<td>Transaction</td>
<td>Read, Create, Update, Delete, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Prepayment Application</td>
<td>vendoreprepaymentapplication</td>
<td>Transaction</td>
<td>Read, Transform, Update, Delete, and Search</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor Return Authorization</td>
<td>vendorreturnauthorization</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Vendor-Subsidiary Relationship</td>
<td>vendorsubsidiaryrelationship</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Website Setup</td>
<td>website</td>
<td>Website</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Weekly Timesheet</td>
<td>timesheet</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Work Order</td>
<td>workorder</td>
<td>Transaction</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Work Order Close</td>
<td>workorderclose</td>
<td>Transaction</td>
<td>Copy and Create Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Work Order Completion</td>
<td>workordercompletion</td>
<td>Transaction</td>
<td>Copy and Create Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Work Order Issue</td>
<td>workorderissue</td>
<td>Transaction</td>
<td>Copy and Create Not Supported</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Workplace</td>
<td>workplace</td>
<td>List</td>
<td>Full</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
Activities

Note: This topic applies to all versions of SuiteScript.

The following activity records are scriptable in SuiteScript:

- Activity
- Event
- Phone Call
- Project Task
- Resource Allocation
- Task
- Work Calendar

Activity

Note: The topic applies to all versions of SuiteScript.

The internal ID for this record is `activity`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The activity record is scriptable in both server and client SuiteScript.

Supported Functions

The activity record can only be searched using SuiteScript.

Event

Note: This topic applies to all versions of SuiteScript.

Events are scheduled activities that are automatically added to your calendar when created.
For help working with this record in the user interface, see the help topic Creating a New Event Record.

The internal ID for this record is **calendarevent**. Note that setting recurring events in SuiteScript is not currently supported.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The event record is scriptable in both server and client SuiteScript.

**Supported Functions**

The event record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

**Usage Notes**

When the Time Tracking feature is enabled, the TimeltemList sublist is available. This list is used to track employee time associated with the phone call, including payroll, billing, and project fields. It is an inline editor sublist.

**Important:** If any case, task, or event record has more than 9500 time entries in the Time Tracking sublist, all cases include a static list of time entries. These static lists are not accessible to scripting. Attempts to script on Time Tracking sublists that are static lists of time entries will not return accurate results.

**Phone Call**

**Note:** This topic applies to all versions of SuiteScript.

Phone calls records are used to document phone call activity. All information submitted for a phone call record is stored on a record in the phone call list, on the customer record who calls and on any contact's records referenced in the call contact list.

For help working with this record in the user interface, see the help topic Working with Phone Calls.

The internal ID for this record is **phonecall**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.
Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The phone call record is scriptable in both server and client SuiteScript.

**Supported Functions**

The phone call record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

**Usage Notes**

When the Time Tracking feature is enabled, the TimeItemList sublist is available. This list is used to track employee time associated with the phone call, including payroll, billing, and project fields. It is an inline editor sublist.

**Important:** If any case, task, or event record has more than 9500 time entries in the Time Tracking sublist, all cases include a static list of time entries. These static lists are not accessible to scripting. Attempts to script on Time Tracking sublists that are static lists of time entries will not return accurate results.

**Project Task**

Note: This topic applies to all versions of SuiteScript.

The project task record can be used to keep track of specific activities and milestones associated with a project.

The internal ID for this record is `projecttask`.

The project task record is available when the Project Management feature is enabled at Setup > Company > Enable Features, on the Company subtab. When the feature is enabled, you can access the project task record in the UI by navigating to an existing project and clicking the New Project Task or New Milestone button.

For help working with this record in the user interface, see the help topic Project Tasks.

Project task records cannot be created as standalone records. Rather, you create a project task for a specific project record, and the task remains attached to that record. For information on working with the project record in SuiteScript, see Project.

See the SuiteScript Records Browser for all internal IDs associated with this record.
For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Project Tasks Versus Milestone Tasks**

Every project task record is designated as either a project task or a milestone task. A project task is used to represent an activity, whereas a milestone task is used to represent a checkpoint in the overall progress of the project.

Although they are the same type of record, a milestone task cannot have values in the estimated work body or sublist fields. Therefore, if you create a project task record and do not include any estimated work, the record is automatically saved as a milestone task. If you do include estimated work, the record is saved as a project task.

These same rules apply to the updating of records as well. In other words:

- To convert a project task into a milestone task, clear the estimated work body field and all values from the Assignees sublist.
- To convert a milestone task into a project task, add a value to the estimated work body field or add at least one record to the Assignees sublist, with a positive value of estimated work.

Note that the estimated work body field is populated by the sum of estimated work listed for Assignees. If you explicitly set a value for the estimated work body field, and you also include Assignees, the value you specify for the body field is overwritten based on the sublist's estimated work values. If you do not include Assignees, you can explicitly assign a value to the estimated work body field.

**Supported Script Types**

The project record is scriptable in server SuiteScript only.

All three user events are supported: `beforeLoad`, `beforeSubmit`, and `afterSubmit`.

**Supported Functions**

The project record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Field Definitions**

When creating new project tasks you must set the Project (company) field to the project/job ID. Project tasks are not standalone records, and therefore must be associated with a specific project.

For other details on body fields and sublist fields, see the SuiteScript Records Browser, which lists all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.
Code Sample

The following example shows how you might create both a project task and a milestone task.

```javascript
// create a project
var project = record.create({
    type: record.Type.JOB,
    isDynamic: true
});
project.setValue({
    fieldId: 'companyName',
    value: 'Reconstruction'
});
project.setValue({
    fieldId: 'subsidiary',
    value: 1
});
var projectId = project.save();

// create a project task
var task = record.create({
    type: record.Type.PROJECT_TASK
});
task.setValue({
    fieldId: 'estimatedwork',
    value: 2
});
task.setValue({
    sublistId: 'title',
    value: 'Remove old furniture'
});
task.setValue({
    sublistId: 'company',
    value: projectId
});
var task1Id = task.save();

// create another task depending on the first one with Finish-To-Start dependency
var task = record.create({
    type: record.Type.PROJECT_TASK
});
task.setValue({
    fieldId: 'estimatedwork',
    value: 5
});
task.setValue({
    fieldId: 'title',
    value: 'Paint walls'
});
task.setValue({
    fieldId: 'company',
    value: projectId
});
task.selectNewLine({
    sublistId: 'predecessor'
});
```
task.setCurrentSublistValue({
    sublistId: 'predecessor',
    fieldId: 'task',
    value: task1Id
});

task.setCurrentSublistValue({
    sublistId: 'predecessor',
    fieldId: 'type',
    value: 'FS'
});

task.commitLine({
    sublistId: 'predecessor'
});

var task2Id = task.save();

// create a milestone

var task2Id = task.save();

var milestoneId = task.save();
Resource Allocation

**Note:** This topic applies to all versions of SuiteScript.

**Important:** For information on the availability of the Resource Allocations feature, please contact your account representative.

The resource allocation record supports reserving an employee's time for a particular project.

The internal ID for this record is `resourceallocation`.

In the UI, you access this record by going to Activities > Scheduling > Resource Allocations. Alternatively, you can view the resource allocations for a specific project through the project record's Resources subtab (Lists > Relationships > Projects, or Lists > Relationships > Jobs).

This record is available only if the Resource Allocations feature has been enabled at Setup > Company > Enable Features, on the Company tab. Note that this option will not be visible unless your account has been provisioned for this feature. For more details, contact your account representative.

For help working with this record in the user interface, see the help topic Resource Allocations.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is scriptable in both client SuiteScript and server SuiteScript.

All three user events are supported: `beforeLoad`, `beforeSubmit`, and `afterSubmit`.

**Supported Functions**

The resource allocation record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Field Definitions**

To create a new resource allocation record, you must reference two existing NetSuite records, as follows:

- `project` — A reference to a project record defined at Lists > Relationships > Projects. Note that in some NetSuite accounts, projects are referred to as jobs.
- `allocationresource` — A reference to an employee record, defined at Lists > Employees > Employees, for which the Project Record option has been checked. The Project Resource box is located on the Human Resources tab of the employee record.

You are also required to provide values for several other fields. Refer to the SuiteScript Records Browser for the internal IDs of all fields associated with this record.
Note also that numberhours and percentoffime are read-only fields. They are returned when you load the record, but they cannot be modified.

For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser.

## Code Samples

The following example shows how to create a resource allocation record:

```javascript
var AMOUNT = "10.0";
var PROJECT1 = "43";
var RESOURCE1 = "45";
var ALLOCATIONTYPE1 = "1"; // Hard
var ALLOCATIONTYPE2 = "2"; // Soft
var ALLOCATIONUNIT1 = "H"; // Hours
var ALLOCATIONUNIT2 = "P"; // Percent of Time
var NOTES1 = "My notes 1";
var STARTDATE1 = "4/20/2013";
var ENDDATE1 = "4/28/2013";

var newRecord = record.create ({
  type: 'resourceallocation'
});

newRecord.setValue ({
  fieldId: 'allocationamount',
  value: AMOUNT
});
newRecord.setValue ({
  fieldId: 'allocationresource',
  value: RESOURCE1
});
newRecord.setValue ({
  fieldId: 'allocationtype',
  value: ALLOCATIONTYPE1
});
newRecord.setValue ({
  fieldId: 'allocationunit',
  value: ALLOCATIONUNIT1
});
newRecord.setValue ({
  fieldId: 'startdate',
  value: STARTDATE1
});
newRecord.setValue ({
  fieldId: 'notes',
  value: NOTES1
});
newRecord.setValue ({
  fieldId: 'project',
  value: PROJECT1
});
newRecord.setValue ({
  fieldId: 'enddate',
  value: ENDDATE1
});
```
Task

**Note:** This topic applies to all versions of SuiteScript.

Tasks are activities that need to be completed. Use the task record to add new tasks for individuals, companies or contacts and to modify those records.

For help working with this record in the user interface, see the help topic Creating a Project Task Record.

The internal ID for this record is `task`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The task record scriptable in both client SuiteScript and server SuiteScript.

**Supported Functions**

The resource allocation record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

When the Time Tracking feature is enabled, the TimeItemList sublist is available. This list is used to track employee time associated with the task, including payroll, billing, and project fields. It is an inline editor sublist.

**Important:** If any case, task, or event record has more than 9500 time entries in the Time Tracking sublist, all cases include a static list of time entries. These static lists are not accessible to scripting. Attempts to script on Time Tracking sublists that are static lists of time entries will not return accurate results.

**Work Calendar**

**Note:** This topic applies to all versions of SuiteScript.

You can set up work calendars to track and manage the work capacity for employees and vendors you assign as resources on projects. Knowing the work capacity for each employee helps you to schedule resources for project tasks.
Also, any employee or vendor you plan to assign as a project resource must have a work calendar assigned on their record.

For help working with this record in the user interface, see the following topics:

- Project Resource Work Calendars
- Setting Up a Work Calendar
- Assigning a Resource Work Calendar

The internal ID for this record is `workcalendar`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Entities**

**Note:** This topic applies to all versions of SuiteScript.

**Entity Record Restrictions for Scripting on Administrator Role**

This section describes entity record restrictions for scripting on the Administrator and Full Access roles. Note that these restrictions apply to all versions of SuiteScript and all NetSuite environments (see the help topic Understanding NetSuite Account Types).

You cannot perform these actions with SuiteScript:

- Create an entity record where any role is set to Administrator or Full Access
- Delete an entity record where any role is set to Administrator or Full Access
- Edit an entity record so that the entity gains an Administrator or Full Access role
- Edit an entity record so that the entity loses an Administrator or Full Access role
- Edit the password or email field value on an entity record where any role is set to Administrator or Full Access

These restrictions contribute to improved security. Scripts that violate them throw an error message: `Script Security Violation: Unauthorized attempt to <operation> entity with <role> role by SuiteScript!`

**Important:** As of 2019.1, changes have been made to the Full Access role, in preparation for its deprecation. For details, see Plan to Deprecate the Full Access Role in the 2019.1 Release Notes.

**Entity Record Types**

The following entity records are scriptable in SuiteScript:
You can use the bonus record to award different types of bonuses to your employees. To use bonus records, the Compensation Tracking feature must be enabled in your account. For help working with this record in the user interface, see the help topic Recording Base Pay Compensation for an Employee.

The internal ID for this record is `bonus`.

See `Bonus Record Action` for the action associated with this record. For more information about actions and macros, see the help topic `Overview of Record Action and Macro APIs`.

See the `SuiteScript Records Browser` for all internal IDs associated with this record.
Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The bonus record is scriptable in both client and server SuiteScript.

Supported Functions

The following SuiteScript functionality is supported:
- Read
- Create
- Update
- Copy
- Delete
- Search

Usage Notes

If a payroll item is associated with the bonus type for the bonus record:
- The update function is not available.
- SuitePeople U.S. Payroll determines the status of the bonus. Workflows cannot see these statuses, so you should not use status changes as event triggers.

Script Sample

The following sample shows how to award a bonus.

```javascript
require(['N/record'], function (record) {
    var employeeId = 659;     // id of selected employee
    var bonusTypeId = 1;      // id of previously created bonus type NOTE: bonus type must have same subsidiary as selected employee
    var percentage = 25;      // set percentage of employee salary for given bonus
    var bonusStatus = 'none'; // bonus status ('none', ready_to_pay', 'being_processed', 'paid', 'cancelled')
    var awardDate = new Date('12/24/2020'); // date in MM/DD/YYYY format

    var bonus = record.create({
        type: record.Type.BONUS
    });
});
```
bonus.setValue({
  fieldId: 'bonusemployee',
  value: employeeId
});
bonus.setValue({
  fieldId: 'bonustype',
  value: bonusTypeId
});
bonus.setValue({
  fieldId: 'bonusawarddate',
  value: awardDate
});
bonus.setValue({
  fieldId: 'bonusstatus',
  value: bonusStatus
});
bonus.setValue({
  fieldId: 'bonusamountrcentage',
  value: percentage
});

bonus.save();

### Bonus Record Action

cancel

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Exclude from Payroll</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Description</strong></td>
<td>Excludes the existing bonus from the upcoming payroll run. The action.ALL_QUALIFIED_INSTANCES constant is currently not supported.</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td>void</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2020.1</td>
</tr>
</tbody>
</table>

### Parameters

See the help topic `action.execute(options)` for details about parameters required for the execution of any action.

### Bonus Cancel Action Syntax

```javascript
require(['N/action'], function(action) {
  // action & action list loading
  var actionList = action.find({
    recordType: 'bonus'
  });
});
```
```javascript
var actionObj = action.get({
  recordType: 'bonus',
  id: 'cancel'
});

// action execution
var result = actionObj.execute({
  recordId: 1
});

var result = action.execute({
  recordType: 'bonus',
  id: 'cancel',
  params: {
    recordId: 2
  }
});
```

**Bonus Type**

*Note:* This topic applies to all versions of SuiteScript.

You can use the bonus type record to define different types of bonuses within your organization's compensation plans. For example, you might have holiday bonuses, sign-on bonuses, or merit bonuses. To use bonus records, the Compensation Tracking feature must be enabled in your account. For help working with this record in the user interface, see the help topic Recording Base Pay Compensation for an Employee.

The internal ID for this record is `bonustype`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

*Note:* For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The bonus type record is scriptable in both client and server SuiteScript.

**Supported Functions**

The following SuiteScript functionality is supported:

- Read
Script Sample

The following sample shows how to create a bonus type record.

```javascript
require(['N/record'], function (record) {
  var bonusType = record.create({
    type: record.Type.BONUS_TYPE
  });
  bonusType.setValue({
    fieldId: 'name',
    value: 'Annual Bonus'
  });
  bonusType.setValue({
    fieldId: 'subsidiary',
    value: 2
  });
  bonusType.save();
});
```

Competitor

**Note:** This topic applies to all versions of SuiteScript.

You can create competitor records to track how other businesses in your industry impact your sales.

If you use NetSuite or NetCRM and have the Opportunities feature enabled, selecting a competitor on an opportunity record lets everyone working with that opportunity know who they are competing with. By being aware of the strengths and weaknesses of the competition, sales reps can make educated offers and close more deals.

For help working with this record in the user interface, see the help topics Competitors and Opportunity Records.

The internal ID for this record is `competitor`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
N/record Module

Supported Script Types

The competitor record is scriptable in both client and server SuiteScript.

Supported Functions

The competitor record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Contact

Contacts represent people or companies that you deal with in the daily activity of your business. Use the contact record to create, modify, or delete contacts and associate a contact to a parent record.

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the user interface, see the help topic Contacts.

The internal ID for this record is contact.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The contact record is scriptable in both client and server SuiteScript.

Supported Functions

The contact record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Customer

Customer records allow you to track all the following types of information on your customers in one place, depending on the features you have enabled.
For help working with this record in the user interface, see the help topic Customers.

Note: This topic applies to all versions of SuiteScript.

The internal ID for this record is customer.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The customer record is scriptable in both client and server SuiteScript.

Supported Functions

The customer record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes

Notes on Scripting Customer Fields

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>datecreated</td>
<td>Date Created</td>
<td>This is a system-generated field that marks the date the record was created in NetSuite. You cannot change or override this field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tip: If you need to capture “date created” information that is not related to the date the record was created in NetSuite, create a custom field and set it to auto-default to today's date.</td>
</tr>
<tr>
<td>password</td>
<td>Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>password2</td>
<td>Confirm Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts</td>
</tr>
</tbody>
</table>
The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

The Access / contactroles sublist is included in the customer record on the Access subtab. It is an inline editor subtab. The Access / contract roles sublist is related to the Contact / contactroles sublist. For details, see the contactroles sublist in the SuiteScript Records Browser.

The Time Tracking sublist is included with the customer record. It is an inline editor sublist.

**Important:** If any case, task, or event record has more than 9500 time entries in the Time Tracking sublist, all cases include a static list of time entries. These static lists are not accessible to scripting. Attempts to script on Time Tracking sublists that are static lists of time entries will not return accurate results.

### Notes on Scripting Customer Sublists

You can update the contactaccessroles sublist to provide Customer Center access to contacts. You can provide access to contacts that already exist in NetSuite and that have already been attached to a customer that already exists in NetSuite. The workflow is as follows: 1) Add customer. 2) Add contacts. 3) Attach contacts to customer. 4) Update customer with contact access information.

The fields in this sublist map to the fields on the Access subtab in the UI. These fields include: a Boolean field that indicates whether a contact has access to NetSuite, contact name key field, email address used to log in to NetSuite, password used to log in to NetSuite, NetSuite role (Customer Center), and a Boolean field that indicates whether the contact should receive a notification email when access changes are made. If this Notify field is set to true, an email is sent.

### Transform Types

In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.
Customer Status

**Note:** This topic applies to all versions of SuiteScript.

The customer status record describes a lead, prospect, or a customer’s stage in the sales cycle. To create a new customer status record, go to Setup > Sales > Setup Tasks > Customer Statuses > New.

For help working with this record in the user interface, see the help topic Customers.

The internal ID for this record is `customerstatus`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The customer status record is scriptable in both client and server SuiteScript.

**Supported Functions**

The customer status record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Employee

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the user interface, see the help topic Employee Information Management.

The internal ID for this record is `employee`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
**N/record Module**

**Supported Script Types**

The employee record is scriptable in both client and server SuiteScript.

**Supported Functions**

The employee record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

**Usage Notes**

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>password2</td>
<td>Confirm Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
</tbody>
</table>

**Scripting when Advanced Employee Permissions is Enabled**

When the Advanced Employee Permissions feature is enabled keep the following in mind:

- Before submit scripts can be run, either as the role that has manipulated the record or as an administrator, the script only has access to the information that has been submitted from the browser. In the case of an employee record being edited, the before submit script only has access to the fields or sublists the role editing the record is permitted to edit. Therefore, you cannot assume that all fields and sublists on the employee record are available in a before submit script.

- After submit scripts can be run as administrator, which means complete access to an employee record. When the Advanced Employee Permissions feature is enabled, perform actions, such as setting the value of a field, as a role that is known to have the appropriate level of access or as an administrator.

For information about this feature, see the help topic [Advanced Employee Permissions](#).

**Transform Types**

In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

<table>
<thead>
<tr>
<th>Target Record Type</th>
<th>Target Record Internal ID</th>
<th>Field Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expense Report</td>
<td>expensereport</td>
<td></td>
</tr>
</tbody>
</table>
Employee Status

**Note:** This topic applies to all versions of SuiteScript.

Employee status values can be used to categorize employees’ work status. Example values include part-time, full-time, temporary, regular. These values display on the Human Resources subtab of employee records.

For help working with this record in the user interface, see the help topic Setting Up Employee Related Lists.

The internal ID for this record is `employeestatus`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The employee status record is scriptable in both client and server SuiteScript.

**Supported Functions**

The following SuiteScript functionality is supported:

- Read
- Create
- Update
- Copy
- Delete
- Search

**Script Sample**

The following sample shows how to create an employee status record.

```javascript
require(['N/record'], function (record) {
```

SuiteScript Records Guide
```javascript
var myRecord = record.create({
  type: record.Type.EMPLOYEE_STATUS
});
myRecord.setValue({
  fieldId: 'name',
  value: Active
});
myRecord.save();
```
**Script Sample**

The following sample shows how to create an employee type record.

```javascript
require(['N/record'], function (record) {
    var myRecord = record.create({
        type: record.Type.EMPLOYEE_TYPE
    });
    myRecord.setValue({
        fieldId: 'name',
        value: 'Exempt'
    });
    myRecord.save();
});
```

**Entity**

- **Note:** This topic applies to all versions of SuiteScript.

The internal ID for this record is `entity`.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The entity record is only exposed in search.

**Supported Functions**

Only search is supported for the entity record.

**Generic Resource**

- **Note:** This topic applies to all versions of SuiteScript.

Generic resource records can be used as placeholders when project managers and resource managers are planning a project in NetSuite. This feature enables resource allocations and project task assignments to be made when a specific resource may not yet be identified.

For help working with this record in the user interface, see the help topic Generic Resources.

The internal ID for this record is `genericresource`.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Use a generic resource as a placeholder for resource allocation and project task assignment. This record is primarily used when a specific resource is not available. See the help topic Generic Resources for additional information.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The generic resource record is supported in all client and server-side scripts.

### Supported Functions

The following SuiteScript functionality is supported:

- Read
- Create
- Edit
- Delete
- Search

**Note:** Copy is not supported.

### Job

**Note:** This topic applies to all versions of SuiteScript.

The Job Management feature must be enabled to script with this record.

The internal ID for this record is `job`.

For help working with this record in the user interface, see the help topic Managing Jobs.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types

The job record is scriptable in both client and server SuiteScript.

Supported Functions

The job record can be read, created, updated, searched, and deleted using SuiteScript. It cannot be copied or transformed.

Usage Notes

The Assigned Employees sublist and Job Requisition sublist are not available to SuiteScript.

Code Sample

```javascript
function beforeLoad(type, form, request){
    // rec in all following statements has been previous defined (created/loaded) as a Job record
    rec.setValue(
        {fieldId: 'title',
        value: 'abc'});
    rec.setValue(
        {fieldId: 'description',
        value: 'test description'});
    rec.setValue(
        {fieldId: 'isinactive',
        value: 'F'});
}
```

Job Requisition

**Note:** This topic applies to all versions of SuiteScript.

The Job Requisitions feature must be enabled to script with this record.

The internal ID for this record is `jobrequisition`.

For help working with this record in the user interface, see the help topic Job Requisitions.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- **N/record Module**

**Supported Script Types**

The job requisition record is scriptable in both client and server SuiteScript.

**Supported Functions**

The job requisition record can be read, created, updated, searched, and deleted using SuiteScript. It cannot be copied or transformed.

**Code Samples**

```javascript
function beforeLoad(type, form, request){
    // rec in all following statements has been previous defined (created/loaded) as a Job Requisition record
    rec.setValue({
        fieldId: 'title',
        value: 'abc'
    });
    rec.setValue({
        fieldId: 'description',
        value: 'test description'
    });
    rec.setValue({
        fieldId: 'isinactive',
        value: 'F'
    });
}

function beforeSubmit(type, form, request){
    rec.setValue({
        fieldId: 'subsidiary',
        value: '1'
    });
    rec.setValue({
        fieldId: 'department',
        value: '2'
    });
    rec.setValue({
        fieldId: 'location',
        value: '3'
    });
}
```

**Lead**

**Note:** This topic applies to all versions of SuiteScript.

Leads are companies or individuals who represent potential customers. In NetSuite, leads are the first step in the sales cycle that progresses to prospect and then to customer.

For help working with this record in the user interface, see the help topic [Lead Management Overview](#).
The internal ID for this record is **lead**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The lead record is scriptable in both client and server SuiteScript.

### Supported Functions

The lead record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

### Usage Notes

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Body Fields</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>datecreated</td>
<td>Date Created</td>
<td>This is a system-generated field that marks the date the record was created in NetSuite. You cannot change or override this field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Tip: If you need to capture “date created” information that is not related to the date the record was created in NetSuite, create a custom field and set it to auto-default to today's date.</td>
</tr>
<tr>
<td>password</td>
<td>Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>password2</td>
<td>Confirm Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
</tbody>
</table>

**Search Filters and Search Columns**

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccnumber</td>
<td>Credit Card Number</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
</tbody>
</table>

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

The Access / contactroles sublist is included in the customer record on the Access subtab. It is an inline editor subtab. The Access / contract roles sublist is related to the Contact / contactroles sublist. For details, see the contactroles sublist in the [SuiteScript Records Browser](#).
The Time Tracking sublist is included with the lead record. It is an inline editor sublist.

**Important:** If any case, task, or event record has more than 9500 time entries in the Time Tracking sublist, all cases include a static list of time entries. These static lists are not accessible to scripting. Attempts to script on Time Tracking sublists that are static lists of time entries will not return accurate results.

### Transform Types

In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

<table>
<thead>
<tr>
<th>Target Record Name</th>
<th>Target Record Internal ID</th>
<th>Field Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opportunity</td>
<td>opportunity</td>
<td></td>
</tr>
</tbody>
</table>

### Other Name

**Note:** This topic applies to all versions of SuiteScript.

The list of other name records is a collection of records for people or companies who are not vendors, customers or employees. This enables you to keep records for other people or companies if you write checks to or receive deposits from them.

For example, your company might donate money to a favorite charity, so you create an other name record for the charity. You might also list your owners and partners here if they contribute or withdraw equity.

For help working with this record in the user interface, see the help topic Other Name Records.

The internal ID for this record is `othername`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The other name record is scriptable in server SuiteScript only.

### Supported Functions

The lead record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Partner

**Note:** This topic applies to all versions of SuiteScript.

A partner is a company you have a business agreement with who isn't a customer or a vendor.

For help working with this record in the user interface, see the help topic Managing Partners.

The internal ID for this record is `partner`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The partner record is scriptable in both client and server SuiteScript.

**Supported Functions**

The partner record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

**Usage Notes**

**Notes on Scripting Partner Fields**

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>password2</td>
<td>Confirm Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
</tbody>
</table>

**Notes on Scripting Partner Sublists**

You can update the contactaccessroles sublist to provide Partner Center access to contacts. You can provide access to contacts that already exist in NetSuite and that have already been attached to a partner...
that already exists in NetSuite. The workflow is as follows: 1) Add partner. 2) Add contacts. 3) Attach contacts to partner. 4) Update partner with contact access information.

The fields in this sublist map to the fields on the Access subtab in the UI. These fields include: a Boolean field that indicates whether a contact has access to NetSuite, contact name key field, email address used to log in to NetSuite, password used to log in to NetSuite, NetSuite role (Partner Center), and a Boolean field that indicates whether the contact should receive a notification email when access changes are made. If this Notify field is set to true, an email is sent.

The Access / contactroles sublist is included in the customer record on the Access subtab. It is an inline editor subtab. The Access / contract roles sublist is related to the Contact / contactroles sublist. For details, see the contactroles sublist in the SuiteScript Records Browser.

Project

Note: This topic applies to all versions of SuiteScript.

Use the project record to manage company initiatives.

To use the project record, you must have the Projects feature enabled at Setup > Company > Enable Features, on the Company tab. If you plan to do advanced project tracking, you must also enable Project Management. If you do not see the Project Management check box, your company must first purchase the Project Management add-on from NetSuite.

To access the project record in the UI, choose Lists > Relationships > Projects (or Jobs).

For help working with this record in the user interface, see .

The internal ID for this record is job.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The project record is scriptable in both client and server SuiteScript.

Supported Functions

The project record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Field Definitions

Note that the datecreated body field is a system-generated field that marks the date the record was created in NetSuite. You cannot change or override this field. If you need to capture "date created" information that is not related to the date the record was created in NetSuite, create a custom field and set it to auto-default to today's date.
For more details on available fields, see the SuiteScript Records Browser, which lists all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

### Usage Notes

The Time Tracking sublist is included with the project (job) record. It is an inline editor sublist.

**Important:** If any case, task, or event record has more than 9500 time entries in the Time Tracking sublist, all cases include a static list of time entries. These static lists are not accessible to scripting. Attempts to script on Time Tracking sublists that are static lists of time entries will not return accurate results.

### Adding a Resource With Multiple Roles

When using SuiteScript, the behavior of the Resources sublist differs slightly from the behavior in the UI. Specifically, in the UI, each line must have a unique value in the Name field. Further, in the UI, you can specify more than one role for each resource.

With SuiteScript, if you want to add a resource that has two different roles, you set up your code as if you are adding two sublist records for that resource -- one for each role. The jobresource values are not required to be unique. The following example illustrates this technique. In this example, only two unique resources are being added, but because one resource has two roles, that resource is represented twice.

```javascript
proj = record.create({
  type: record.Type.JOB
});
proj.setValue({
  fieldId: 'companyname',
  value: 'Launch'
});
proj.setValue({
  fieldId: 'subsidiary',
  value: '1'
});
proj.selectNewLine({
  sublistId: 'jobresources'
});
proj.setCurrentSublistText({
  sublistId: 'jobresources',
  fieldId: 'jobresource',
  text: 'Employee Resource 1'
});
proj.setCurrentSublistText({
  sublistId: 'jobresources',
  fieldId: 'role',
  text: 'Staff'
});
```
Note that after you complete the add operation, the Resources sublist in the UI looks the same as it would if you had manually added one line for the resource, with multiple roles specified on that line, as shown in the illustration above.

Scripting Projects with Advanced Revenue Management

The following table lists the scriptable fields associated with the Project record and advanced revenue management:

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Internal ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accounting Period</td>
<td>List/Record</td>
<td>accountingperiod</td>
</tr>
<tr>
<td>Comments</td>
<td>Text Area</td>
<td>comments</td>
</tr>
</tbody>
</table>
Before you begin working with advanced revenue management programmatically, see the help topic Setup for Advanced Revenue Management.

For help working with this record in the user interface, see the help topic Advanced Revenue Management for Projects.

Project Status

*Note:* This topic applies to all versions of SuiteScript.

The project status record indicates the progress of the project.

You can create new statuses at Setup > Accounting > Accounting Lists > New > Project Status.

To use the project status and Project records, you must have the Projects feature enabled at Setup > Company > Enable Features, on the Company tab. If you plan to do advanced project tracking, you must also enable Project Management. If you do not see the Project Management check box, your company must first purchase the Project Management add-on from NetSuite.

For help on how this record is used with the Project record in the user interface, see the help topic Creating a Project Record.

The internal ID for this record is `jobstatus`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

*Note:* For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The project status record is scriptable in both client and server SuiteScript.

Supported Functions

The project status record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Project Template

*Note:* This topic applies to all versions of SuiteScript.

Project Templates enable you to create project records in NetSuite for projects your business performs repeatedly.
To use the project template and Project records, you must have the Projects feature enabled at Setup > Company > Enable Features, on the Company tab. If you plan to do advanced project tracking, you must also enable Project Management. If you do not see the Project Management check box, your company must first purchase the Project Management add-on from NetSuite.

For help working with this record in the user interface, see the help topic Project Templates.

The internal ID for this record is projecttemplate.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Use a project template as a standard starting point for projects and project items. Each record instance is reusable. See the help topic Project Templates for information on using project templates in the UI.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The project template record is supported in all client and server-side scripts.

**Supported Functions**

The following SuiteScript functionality is supported:

- Read
- Create
- Edit
- Delete
- Search

**Note:** Copy is not supported.

**Project Type**

**Note:** This topic applies to all versions of SuiteScript.

To use the project type and Project records, you must have the Projects feature enabled at Setup > Company > Enable Features, on the Company tab. If you plan to do advanced project tracking, you must also enable Project Management. If you do not see the Project Management check box, your company must first purchase the Project Management add-on from NetSuite.

For help working with this record in the user interface, see the help topic Basic Projects.

The internal ID for this record is jobtype.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The project type record is scriptable in both client and server SuiteScript.

**Supported Functions**

The project type record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

**Prospect**

Note: This topic applies to all versions of SuiteScript.

Prospect records enable you to track all the information you need to convert a prospect into a customer.

For help working with this record in the user interface, see the help topic Prospects.

The internal ID for this record is `prospect`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The prospect record is scriptable in both client and server SuiteScript.

**Supported Functions**

The prospect record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.
Usage Notes

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>password2</td>
<td>Confirm Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
</tbody>
</table>

Search Filters and Search Columns

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccnumber</td>
<td>Credit Card Number</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
</tbody>
</table>

Important: If any case, task, or event record has more than 9500 time entries in the Time Tracking sublist, all cases include a static list of time entries. These static lists are not accessible to scripting. Attempts to script on Time Tracking sublists that are static lists of time entries will not return accurate results.

Transform Types

In the NetSuite Help Center, see the help topic record.transform(options) for examples on how to transform records.

<table>
<thead>
<tr>
<th>Target Record Name</th>
<th>Target Record Internal ID</th>
<th>Field Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimate/Quote</td>
<td>estimate</td>
<td></td>
</tr>
<tr>
<td>Opportunity</td>
<td>opportunity</td>
<td></td>
</tr>
<tr>
<td>Sales Order</td>
<td>salesorder</td>
<td></td>
</tr>
</tbody>
</table>

Time-Off Management

Important: This topic applies to all versions of SuiteScript.

The following record types are related to the Time-Off Management feature:
Time-Off Change

The internal ID for this record is `timeoffchange`.

For help working with this record in the user interface, see the help topic Viewing an Employee's Time-Off Balance.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The time-off change record is scriptable in server SuiteScript only.

Note that triggering of user events is not supported for this record.

Supported Functions

The time-off change record is not fully scriptable. Transform and copy are not permitted.

Code Samples

The following example shows how to create a time-off change.

```javascript

... 
var timeOffChange = record.create({
    type: record.Type.TIME_OFF_CHANGE
});
timeOffChange.setValue({
    fieldId: 'timeofftype',
    value: '1'                // ID of time-off type. In this example, ID of 1 is used.
});
timeOffChange.setValue({
    fieldId: 'description',
    value: 'Time-Off Type Name'
});
```

SuiteScript Records Guide
timeOffChange.setValue(
    {  
      fieldId: 'dateapplied',
      value: '12/25/2016'
    });
timeOffChange.setValue(
    {  
      fieldId: 'amount',
      value: '1'
    });
timeOffChange.setValue(
    {  
      fieldId: 'timeoffunit',
      value: '1' // ID of unit from drop-down
    });
var id = timeOffChangeReq.save();
...

**Time-Off Plan**

The internal ID for this record is **timeoffplan**.

For help working with this record in the user interface, see the help topic **Creating a Time-Off Plan**.

See the **SuiteScript Records Browser** for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic **Working with the SuiteScript Records Browser** in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- **SuiteScript 2.0 Scripting Records and Subrecords**
- **SuiteScript 2.0 Custom Forms**
- **N/record Module**

**Supported Script Types**

The time-off plan record is scriptable in server SuiteScript only.

Note that triggering of user events is not supported for this record.

**Supported Functions**

The time-off plan record can be read, created, updated, searched, and deleted using SuiteScript. It cannot be copied or transformed.

**Code Samples**

The following example shows how to create a time-off plan and modify an existing time-off plan.

```javascript
...
//Create the Time-Off Plan record
var timeOffPlan = record.create({
  type: record.Type.TIME_OFF_PLAN
});
...
```javascript
  timeOffPlan.setValue({
    fieldId: 'name',
    value: 'new Time-Off Plan'
  });
  timeOffPlan.setValue({
    fieldId: 'isinactive',
    value: 'F'
  });
  timeOffPlan.setValue({
    fieldId: 'includefutureaccruals',
    value: 'F'
  });
  timeOffPlan.setValue({
    fieldId: 'subsidiary',
    value: '1'              // Use ID of subsidiary, location, department, or class. In this example, ID of 1 is used.
  });
  timeOffPlan.setValue({
    fieldId: 'location',
    value: '1'
  });
  timeOffPlan.setValue({
    fieldId: 'department',
    value: '1'
  });
  timeOffPlan.setValue({
    fieldId: 'class',
    value: '1'
  });
  timeOffPlan.setValue({
    fieldId: 'startmonth',
    value: '1'
  });

  var id = timeOffPlan.save();

  // Modify existing Time-Off Plan record
  var timeOffPlan = record.load({
    type: record.Type.TIME_OFF_PLAN,
    id: id
  });
  timeOffPlan.setValue({
    fieldId: 'isinactive',
    value: 'T'                // Inactivate plan
  });
  timeOffPlan.save();

  ...
```

**Time-Off Request**

The internal ID for this record is `timeoffrequest`.

For help working with this record in the user interface, see the help topics Submitting Time-Off Requests and Approving or Rejecting Time-Off Requests.
See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Note:** To submit time-off requests, employees must have a start date and time-off plan assigned to them on the employee record.

**Supported Script Types**

The time-off request record is scriptable in server SuiteScript only.

Note that triggering of user events is not supported for this record.

**Supported Functions**

The time-off request record is not fully scriptable. Transform and copy are not permitted.

**Code Samples**

The following example shows how to create and approve a time-off request.

```javascript
...
//Create Time-Off Request
var timeOffRequest = record.create({
  type: record.Type.TIME_OFF_REQUEST
});
timeOffRequest.setValue({
  fieldId: 'employee',
  value: '25'
});
timeOffRequest.setValue({
  fieldId: 'startdate',
  value: '12/25/2016'
});
timeOffRequest.setValue({
  fieldId: 'enddate',
  value: '12/26/2016'
});
timeOffRequest.selectNewLine({
  sublistId: 'detail'
});
timeOffRequest.setCurrentSublistValue({
  sublistId: 'detail',
  fieldId: 'timeoffdate',
  value: '12/25/2016'
});
```
### Time-Off Rule

The internal ID for this record is `timeoffrule`.

For help working with this record in the user interface, see the help topic Creating a Time-Off Rule.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

> **Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

### Supported Script Types

The time-off rule record is scriptable in server SuiteScript only.

Note that triggering of user events is not supported for this record.

### Supported Functions

The time-off rule record is not fully scriptable. Copy and search are not permitted.
Code Samples

The following code snippets show how to create a time-off rule.

```javascript
//Create Time-Off Rule to add to existing Time-Off Plan record.
var timeOffRule = record.create({
    type: record.Type.TIME_OFF_RULE
});
timeOffRule.setValue({
    fieldId: 'openingbalancefornewemployees',
    value: 'T'
});
timeOffRule.setValue({
    fieldId: 'entitlement',
    value: '1'
});
timeOffRule.setValue({
    fieldId: 'maximumtenure',
    value: '1'
});
timeOffRule.setValue({
    fieldId: 'minimumtenure',
    value: '0'
});
timeOffRule.setValue({
    fieldId: 'maximumtenureunit',
    value: '1'                  // ID of unit available in drop-down. In this example, ID of 1 is used.
});
timeOffRule.setValue({
    fieldId: 'minimumtenureunit',
    value: '1'
});
timeOffRule.setValue({
    fieldId: 'entitlementunit',
    value: '1'
});
timeOffRule.setValue({
    fieldId: 'timeofftype',
    value: '1'                  // Use ID of Time-Off Type. In this example, ID of 1 is used.
});
timeOffRule.setValue({
    fieldId: 'timeoffplan',
    value: '1'                  // Use ID of Time-Off Plan you want to assign a rule to. In this example, ID of 1 is used.
});
var id = timeOffRule.save();
```

Time-Off Type

The internal ID for this record is `timeofftype`. 
For help working with this record in the user interface, see the help topic Creating a Time-Off Type.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The time-off type record is scriptable in server SuiteScript only.

Note that triggering of user events is not supported for this record.

**Supported Functions**

The time-off type record is not fully scriptable. Transform and copy are not permitted.

**Code Samples**

The following example shows how to create a time-off type.

```javascript
//Create Time-Off Type
var timeOffType = record.create(
    {type: record.Type.TIME_OFF_TYPE});
timeOffType.setValue(
    {fieldId: 'name',
     value: 'Time-Off Type Name'});
timeOffType.setValue(
    {fieldId: 'displayname',
     value: 'Time-Off Type Name'});
timeOffType.setValue(
    {fieldId: 'istrackonly',
     value: 'T'});
timeOffType.setValue(
    {fieldId: 'isautogeneratetimeentry',
     value: 'T'});
timeOffType.setValue(
    {fieldId: 'minimumincrement',
     value: '1'});
```
Vendor

Note: This topic applies to all versions of SuiteScript.

A vendor is a company or person you purchase goods and services from. Vendor records track information about your vendors and enable you to view past transactions and communications with them.

For help working with this record in the user interface, see the help topic Vendor Records Overview.

The internal ID for this record is vendor.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The vendor record is scriptable in both client and server SuiteScript.

Supported Functions

The vendor record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body Fields</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table: Field Internal ID, Field UI Label, Note

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>password</td>
<td>Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>password2</td>
<td>Confirm Password</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
</tbody>
</table>

**Transform Types**

In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

<table>
<thead>
<tr>
<th>Target Record Name</th>
<th>Target Record Internal ID</th>
<th>Field Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purchase Order</td>
<td>purchaseorder</td>
<td></td>
</tr>
<tr>
<td>Vendor Bill</td>
<td>vendorbill</td>
<td></td>
</tr>
<tr>
<td>Vendor Payment</td>
<td>vendorpayment</td>
<td></td>
</tr>
</tbody>
</table>

**Items**

- Using Item Records in SuiteScript
- Pricing Sublist / Pricing Matrix
- Assembly Item
- Description Item
- Discount Item
- Download Item
- Gift Certificate Item
- Inventory Item
- Item Search
- Item Group
- Kit Item
- Lot Numbered Assembly Item
- Lot Numbered Inventory Item
- Markup Item
- Non-Inventory Part
- Other Charge Item
- Payment Item
- Reallocate Items

**Note:** This topic applies to all versions of SuiteScript.
■ Serialized Assembly Item
■ Serialized Inventory Item
■ Service
■ Shipping Item
■ Subscription Plan
■ Subtotal Item
■ Supply Chain Snapshot Simulation

Using Item Records in SuiteScript

Note: This topic applies to all versions of SuiteScript.

This section includes the following topics:

■ Loading Item Types
■ Filtering Items by Type
■ Advanced Revenue Management Scripting with Items

For information about working with items in the UI, see the help topics Using Item Records and Item Types.

Loading Item Types

When using record.load(options), you can:

■ set the type parameter to record.Type.INVENTORY_ITEM to load the following types of item records: inventoryitem, lotnumberedinventoryitem, serializedinventoryitem
■ set the type parameter to record.Type.ASSEMBLY_ITEM to load the following types of item records: assemblyitem, lotnumberedassemblyitem, serializedassemblyitem

Filtering Items by Type

The following are valid search filter item type IDs. Note that the item filter IDs are case-sensitive.

<table>
<thead>
<tr>
<th>Item Type IDs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Assembly</td>
<td>Markup</td>
</tr>
<tr>
<td>Description</td>
<td>NonInvtPart</td>
</tr>
<tr>
<td>Discount</td>
<td>OthCharge</td>
</tr>
<tr>
<td>DwnLdItem</td>
<td>Payment</td>
</tr>
<tr>
<td>EndGroup</td>
<td>Service</td>
</tr>
<tr>
<td>GiftCert</td>
<td>ShipItem</td>
</tr>
<tr>
<td>Group</td>
<td>Subtotal</td>
</tr>
<tr>
<td>InvPart</td>
<td>TaxGroup</td>
</tr>
<tr>
<td>Kit</td>
<td>TaxItem</td>
</tr>
</tbody>
</table>

SuiteScript Records Guide
To use these IDs:

1. Create a script that will search for items of a specific type or types (for example, search for all non-inventory items).
2. Next, see any of the valid SuiteScript item type IDs.

Sample Code

```javascript
//Create a script that will search for all non-inventory part items
function searchnoninventorypart()
{
  var searchType = search.Type.NON_INVENTORY_ITEM;
  var mySearch = search.create({
    type: searchType,
    columns: [{
      name: 'internalId'
    }],
    filters: [{
      name: ' ',
      operator: search.Operator.ANYOF,
      values: 'NonInvtPart'
    }]
  });
}
```

Advanced Revenue Management Scripting with Items

The item record (internal ID **item**) contains several additional accounting fields associated with the Advanced Revenue Management feature. The following table lists these scriptable fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Internal ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Revenue Plans On</td>
<td>List/Record</td>
<td>createRevenueplanon</td>
</tr>
<tr>
<td>Item Revenue Category</td>
<td>List/Record</td>
<td>itemrevenuecategory</td>
</tr>
<tr>
<td>Revenue Allocation Group</td>
<td>List/Record</td>
<td>revenueallocationgroup</td>
</tr>
<tr>
<td>Revenue Recognition Rule</td>
<td>List/Record</td>
<td>revenuerecognitionrule</td>
</tr>
</tbody>
</table>

Before you begin working with advanced revenue management programmatically, see the help topic **Setup for Advanced Revenue Management**.

For help working with this record in the UI, see the help topic **Item Configuration for Advanced Revenue Management**.

Pricing Sublist / Pricing Matrix

**Note:** This topic applies to all versions of SuiteScript.

The Pricing sublist is often referred to as a pricing matrix, since there can be multiple prices specified for an item as determined by one or more price levels and one or more quantity levels.

Functionally, the Pricing sublist shares many of the characteristics of **List Sublists**. However, scripting to the Pricing sublist is not like scripting to other sublists in NetSuite. For this reason, it is recommended that...
you read all of the following topics to learn about using SuiteScript on the Pricing sublist. These topics do not need to be read in order, although it is recommended:

- What is the Pricing Matrix?
- Pricing Sublist Feature Dependencies
- Pricing Sublist Internal IDs
- Pricing Sublist Code Sample
- Matrix Sublist APIs and Standard Sublist APIs

**Note:** The screenshots in this section depict the NetSuite UI prior to Version 2010 Release 2.

For general information on item pricing, see these topics in the NetSuite Help Center:

- Item Pricing
- Setting Up Item Pricing
- Using Item Records

To see which records the Pricing sublist appears on, see Records that Include the Pricing Sublist.

**What is the Pricing Matrix?**

Depending on the features enabled in your account, the Pricing sublist on many item records can resemble a matrix of rows and columns of various prices (see figure). To access the price values on a per-row, per-column basis, you must use matrix APIs. These APIs are a subset of the standard APIs that are more commonly used when scripting with other sublists. See Matrix Sublist APIs and Standard Sublist APIs for more information.

**Note:** Non-matrix sublist APIs can also be used on the Pricing sublist. However, they are not used to get or set values considered to be part of the pricing matrix. For information on when to use matrix and non-matrix APIs on the Pricing sublist, see Matrix Sublist APIs and Standard Sublist APIs.

The figure below provides an overview of the rows and columns considered to be the pricing matrix. As previously stated, the configuration of the Pricing sublist greatly depends on the features enabled in your account. However, regardless of the features set, all configurations will have some variation of a row / column matrix layout like the one shown below.
This code snippet shows the kind of values you will typically set when working with price values in the pricing matrix. The internal ID of the Pricing sublist, as well as its field IDs, will change depending on the features enabled in your account.

**Note:** See Pricing Sublist Feature Dependencies and Pricing Sublist Internal IDs for more information.

**Example**

```javascript
record.getMatrixSublistValue({
  sublistId: 'price',
  fieldId: 'price',
  column: 2,
  line: 1
});
```

In this sample you:

1. Specify the sublist internal ID (**price**).
2. Specify the internal ID of the pricing field (which will generally be **price**).

**Important:** Although the UI labels in this figure show field names such as Alternate Price 1, Alternate Price 2, and Online Price, the internal ID for the fieldId parameter is still **price**. The only exception to this is described in Pricing Sublist Field IDs for the currency field.

3. Specify the line number (row) of the price you want to get (in this sample, you are getting the value in row 2 - this is the price for **Alternate Price 1**).
4. Specify the column number you want to get the value for (in this sample, you are getting the value in column 1 for **Alternate Price 1**).

**Price** is typically the internal ID for the fieldId parameter.
Pricing Sublist Feature Dependencies

There are three features that, if enabled or disabled in your account, can affect the overall functionality of the Pricing sublist, its appearance in the UI, and the internal IDs that are referenced in SuiteScript.

You can check which of these features are enabled by looking in the UI, or by calling the `runtime.isFeatureInEffect(options)` method and specifying the feature internal ID.

The features that affect the Pricing sublist are:

- **Multiple Currencies**
- **Multiple Prices**
- **Quantity Pricing**

If none of these features are enabled in your account, then there is no Pricing sublist on the item record, and the field that holds the item price appears on the Basic subtab as Sales Price. The internal ID for this field is `rate`. You do not use sublist methods to set or get values on `rate`. Instead, use field methods.

### Multiple Currencies

This feature allows for item prices to be set in multiple currencies. Separate pricing is specified for each currency. On the Pricing sublist you will see subtabs with the name of the currencies specified in your account (see figure).

**Important:** See Pricing Sublist ID to learn how to determine the internal ID of the Pricing sublist based on whether the Multiple Currencies feature is enabled.

In the UI, you can check if this feature is enabled by looking at the Pricing sublist itself or by going to Setup > Company > Enable Features. On the Company tab, the Multiple Currencies box will be selected if this feature is enabled. Note: Only a NetSuite administrator can enable this feature.

In SuiteScript, you can get the feature status by writing something similar to:

```javascript
var multiCurrency = runtime.isFeatureInEffect({
    feature: 'MULTICURRENCY'
});
```
Multiple Prices

This feature allows different prices to be specified for different conditions or types of customers. This requires that Price Levels are set up. There are a set of standard Price Levels provided by NetSuite, and these can be changed or extended by the customer.

This figure shows the Pricing sublist with the Multiple Prices feature enabled. Notice you can specify multiple prices for the same item.

By comparison, this figure shows the Pricing sublist with the Multiple Prices feature disabled. You can set only one price for the item.

In the UI, you can check if this feature is enabled by looking at the Pricing sublist itself or by going to Setup > Company > Enable Features. On the Transactions subtab, the Multiple Prices box will be selected if this feature is enabled. Note: Only a NetSuite administrator can enable this feature.

In SuiteScript, you can get the feature status by writing something similar to:

```javascript
var multiPrice = runtime.isFeatureInEffect({'feature': 'MULTIPRICE'});
```
Important: See Pricing Sublist Code Sample for more details.

Quantity Pricing

This feature allows the item price to vary based on the quantity of items sold. Specifically, this feature allows different quantity levels to be specified and allows the price to vary at each quantity level.

This figure shows the Pricing sublist with the Quantity Pricing feature enabled.

Note: When the Quantity Pricing feature is enabled, an administrator can specify the number of Qty columns that appear on the Pricing sublist. The following figure shows that four Qty columns have been specified. Set the Qty preference by going to Setup > Accounting > Accounting Preferences > Items & Transactions. In the Maximum # of Quantity-based Price Levels field, specify the number of columns.

Item pricing is determined by values specified in the Qty fields.

By comparison, this figure shows the Pricing sublist with the Quantity Pricing feature disabled. Item prices are not determined by the quantities specified.

No Qty fields exist in the following image:
In the UI, you can check if this feature is enabled by looking at the Pricing sublist itself or by going to Setup > Company > Enable Features. On the Transactions subtab, the Quantity Pricing box will be selected if this feature is enabled. Note: Only a NetSuite administrator can enable this feature.

In SuiteScript, you can get the feature status by writing something similar to:

```javascript
var quantityPricing = runtime.isFeatureInEffect({
    feature: 'QUANTITYPRICING'
});
```

**Important:** See Pricing Sublist Code Sample for more details.

### Pricing Sublist Internal IDs

As discussed in Pricing Sublist Feature Dependencies, the Pricing sublist looks and functions different depending on the features set in your account.

See Pricing Sublist ID for the internal ID of the Pricing sublist depending on features enabled in your account.

See Pricing Sublist Field IDs for all other field IDs associated with this sublist.

### Pricing Sublist ID

In SuiteScript, the internal ID of the Pricing sublist is determined by the features enabled in your NetSuite account.

If the Multiple Currencies feature is not enabled in your account, the internal ID for the Pricing sublist is price. This means that you will set the sublist parameter in APIs such as `Record.getMatrixHeaderField(options)` and `Record.setMatrixHeaderValue(options)` to price.

If Multiple Currencies is enabled, then there are separate Pricing sublists per currency (see figure).

Each currency pricing list will have its own internal ID. For example, the internal ID for the currency called USA will be `price1`. This ID reflects the internal ID of the sublist (`price`) and the internal ID of the USA currency (1).
The internal ID for the Canadian dollar sublist will be **price3**. This reflects the internal ID of the sublist (**price**) and the internal ID of the Canadian dollar currency (**3**).

This figure shows the currencies that have been set in this account. Notice the Internal ID for each currency is the numeric value appended to **price**. When the Multiple Currencies feature is enabled, you can see the internal ID for each currency by going to Lists > Accounting > Currencies.

Based on the internal ID shown in the figure above, you will set the sublistId parameter in APIs such as `Record.getMatrixHeaderField(options)` as follows (fieldId and column parameter values are examples and are not actual valid values):

```
// if scripting on the USA dollar tab:
record.getMatrixHeaderField({
    sublistId: price1,
    fieldId: fldname,
    column: column
  })

// if scripting on the British pound tab:
record.getMatrixHeaderField({
    sublistId: price2,
    fieldId: fldname,
    column: column
  })

// if scripting on the Canadian dollar tab:
record.getMatrixHeaderField({
    sublistId: price3,
    fieldId: fldname,
    column: column
  })

// if scripting on the Euro tab:
record.getMatrixHeaderField({
    sublistId: price4,
    fieldId: fldname,
    column: column
  })
```

For topics related to this one, see Pricing Sublist Feature Dependencies.

**Pricing Sublist Field IDs**

This table provides the internal IDs for all fields associated with the Pricing sublist. Field types are categorized as matrix fields, sublist fields, and body fields.
In SuiteScript, use the IDs that appear in the “Field Internal ID” column for the fieldId values in Sublist APIs and Field APIs.

<table>
<thead>
<tr>
<th>Field UI Label</th>
<th>Field Internal ID</th>
<th>Field Type</th>
<th>Mandatory</th>
<th>Field Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Matrix Fields</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Base Price Qty</td>
<td>price</td>
<td>string</td>
<td>true</td>
<td>The price for that level and that quantity. See Figure 1 - Matrix Fields.</td>
</tr>
<tr>
<td><strong>Sublist Fields</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Default Discount %</td>
<td>discount</td>
<td></td>
<td></td>
<td>See Figure 2 - Sublist Fields. See also Standard Sublist APIs for a code sample that references this internal ID.</td>
</tr>
<tr>
<td>currency</td>
<td></td>
<td></td>
<td></td>
<td>The currency field is a hidden field. It is not visible in the UI. See Figure 2 - Sublist Fields. This field is scriptable only when the Multiple Currencies feature is enabled. The internal IDs for the fieldId parameter in matrix APIs will be price1currency, price2currency, pricecurrency3, etc., to reflect the currency internal ID.</td>
</tr>
<tr>
<td>Price Level</td>
<td>pricelevel</td>
<td></td>
<td></td>
<td>The pricelevel for this price. See Figure 2 - Sublist Fields. See also Standard Sublist APIs for a code sample that references this internal ID.</td>
</tr>
<tr>
<td><strong>Body Fields</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Quantity Pricing Schedule</td>
<td>quantitypricingschedule</td>
<td>select</td>
<td>false</td>
<td>If a quantity pricing schedule has been specified in the UI drop-down, item prices will be calculated for the pricing matrix according to the specified schedule. See Figure 3 - Body Fields. Also note that the value of the quantity pricing schedule sets the value of the Calculate Quantity Discounts (overallquantitypricingtype) field. Use field methods in the N/record Module or N/currentRecord Module module to access this field.</td>
</tr>
<tr>
<td>Calculate Quantity Discounts</td>
<td>overallquantitypricingtype</td>
<td>select</td>
<td>false</td>
<td>Used to determine the quantity amount at the time the item is priced on the order. (This field does not change price settings in the matrix). See Figure 3 - Body Fields.</td>
</tr>
<tr>
<td>Field UI Label</td>
<td>Field Internal ID</td>
<td>Field Type</td>
<td>Mandatory</td>
<td>Field Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>------------</td>
<td>-----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Use Marginal Rates</td>
<td>usemarginalrates</td>
<td>checkbox</td>
<td>false</td>
<td>Used to determine how the quantity discounts are applied at the time the item is priced on the order with a specified quantity. (This field does not change the price settings in the matrix.) See Figure 3 - Body Fields. Use field methods in the N/record Module or N/currentRecord Module module to access this field.</td>
</tr>
<tr>
<td>Pricing Group</td>
<td>pricinggroup</td>
<td>select</td>
<td>false</td>
<td>Used to provide customer-specific pricing. Could affect the pricing at the time the item is placed on the order and associated with a specific customer. See Figure 3 - Body Fields. Use field methods in the N/record Module or N/currentRecord Module module to access this field.</td>
</tr>
</tbody>
</table>

**Figure 1. Figure 1 - Matrix Fields**

![Pricing sublist matrix field.](image)

Pricing sublist matrix field.
Figure 2. Figure 2 - Sublist Fields

Use non-matrix sublist APIs to get or set values for these pricing sublist fields.

Figure 3. Figure 3 - Body Fields

Use standard field APIs to get or set values for Pricing sublist body fields.

Pricing Sublist Code Sample

This sample shows how to determine which pricing-related features are enabled in your account. It then shows how to programmatically determine the internal ID for the Pricing sublist itself, and then check to see if a Quantity Schedule has been applied to the items in this list. The script also shows how to set item prices and quantity levels depending on various conditions set within the pricing matrix.

Note: If your browser is inserting scroll bars in this code sample, maximize your browser window, or expand the main frame that this sample appears in.

// Check the features enabled in the account. See Pricing Sublist Feature Dependencies for // details on why this is important.
var multiCurrency = runtime.isFeatureInEffect({
  feature: 'MULTICURRENCY'
});
var multiPrice = runtime.isFeatureInEffect({
feature: 'MULTIPRICE'
});
var quantityPricing = runtime.isFeatureInEffect({
    feature: 'QUANTITYPRICING'
});

// Set the name of the Price sublist based on which features are enabled and currency type.
// See Pricing Sublist Internal IDs for details on why this is important.
var priceID;
var currencyID = 'EUR';

// Set the ID for the sublist and the price field. Note that if all pricing-related features
// are disabled, you will set the price in the rate field. See Pricing Sublist Feature Dependencies
// for details.
if (multiCurrency === true && multiPrice === false && quantityPricing === false)
    priceID = 'rate';
else {
    priceID = 'price';
    if (multiCurrency === true)
        var internalId = search.create({
            type: search.Type.CURRENCY,
            filters: [{
                name: 'symbol',
                operator: search.Operator.CONTAINS,
                values: currencyID
            }],
        });
    priceID = priceID + internalId;
}

// Check to see if the item is using a Quantity Schedule
// If a Quantity Schedule is used, only the base price needs to be set.
// All other prices will be set according to the schedule.
var itemRecord = record.load({
    type: record.Type.INVENTORY_ITEM,
    id: itemId
});
var qtyPriceSchedule = itemRecord.getValue({
    fieldId: 'quantitypricingschedule'
});

// Set the base price
var basePrice = 100;

// You must select, set, and then commit the sublist line you want to change.
itemRecord.selectLine({
    sublistId: priceID,
    line: 1
});
itemRecord.setCurrentMatrixSublistValue({
    sublistId: priceID,
    fieldId: 'price',
    column: 1,
    value: basePrice
});
itemRecord.commitLine(
    sublistId: priceID
));

// Get the number of columns in the price matrix
// Each column represents a different quantity level
columnCount = itemRecord.getMatrixHeaderCount(
    sublistId: priceID,
    fieldId: 'price'
));

// Set the base price in each quantity of the price matrix for a specific sublist, e.g., currency
for (var j = 1; j <= columnCount; j++){
    itemRecord.selectLine(
        sublistId: priceID,
        line: 1
    );
    itemRecord.setCurrentMatrixSublistValue(
        sublistId: priceID,
        fieldId: 'price',
        column: j,
        value: currencyBasePrice
    );
    itemRecord.commitLine(
        sublistId: priceID
    );
}

// Display the full price matrix for a specific currency as an HTML table

// get the size of the matrix
var quantityLevels = itemRecord.getMatrixHeaderCount(
    sublistId: priceID,
    fieldId: 'price'
));
var priceLevels = itemRecord.getLineCount(
    sublistId: priceID
));
var priceName = "";
var priceNameField = 'pricelevel';
var itemPrice = 0;
var fieldObj = null;

// create an xml table to present the results
var strName = '<table>'
if (quantityLevels > 1){
    strName += '<tr'>
    // write out the quantity levels as the first row
    for (var j = 1; j <= quantityLevels; j++){
        strName += '<td>'
        fieldObj = itemRecord.getMatrixHeaderField(
            sublistId: priceID,
            fieldId: priceNameField,
            column: j
        )
        strName += fieldObj.value
        strName += '</td>
    }
    strName += '</tr>'
    strName += '</table>'
}
Matrix Sublist APIs and Standard Sublist APIs

When writing SuiteScript against the Pricing sublist, you may end up using different types of Sublist APIs. If you want to get or set values in the pricing matrix, you will use matrix APIs (within N/record Module and N/currentRecord Module modules).
If you want to get or set non-matrix fields, you will use all non-matrix sublist or field (within the \texttt{N/record Module} or \texttt{N/currentRecord Module} modules), depending on which fields you are trying to access.

**Note:** See \textit{What is the Pricing Matrix?} for information on the pricing matrix. See \textit{Pricing Sublist Field IDs} for information on the differences among matrix, non-matrix, and body sublist fields.

**Matrix APIs**

The following are considered to be matrix APIs for use on the Pricing sublist. Use these APIs to get or set matrix fields. See \textit{Pricing Sublist Field IDs} to learn which fields are considered matrix fields.

Click the following links to see the API documentation for each matrix API. Also see the figures below for a visual representation for where on the pricing matrix each matrix API executes.

- \texttt{Record.findMatrixSublistLineWithValue(options)}
- \texttt{Record.getCurrentMatrixSublistValue(options)}
- \texttt{Record.getMatrixHeaderCount(options)}
- \texttt{Record.getMatrixHeaderField(options)}
- \texttt{Record.getMatrixHeaderValue(options)}
- \texttt{Record.getMatrixSublistField(options)}
- \texttt{Record.getMatrixSublistValue(options)}
- \texttt{Record.setCurrentMatrixSublistValue(options)}
- \texttt{Record.setMatrixHeaderValue(options)}
- \texttt{Record.setMatrixSublistValue(options)}
- \texttt{CurrentRecord.findMatrixSublistLineWithValue(options)}
- \texttt{CurrentRecord.getCurrentMatrixSublistValue(options)}
- \texttt{CurrentRecord.getMatrixHeaderCount(options)}
- \texttt{CurrentRecord.getMatrixHeaderField(options)}
- \texttt{CurrentRecord.getMatrixHeaderValue(options)}
- \texttt{CurrentRecord.getMatrixSublistField(options)}
- \texttt{CurrentRecord.getMatrixSublistValue(options)}
- \texttt{CurrentRecord.setCurrentMatrixSublistValue(options)}
- \texttt{CurrentRecord.setMatrixHeaderValue(options)}
- \texttt{CurrentRecord.setMatrixSublistValue(options)}
On matrix header fields, use `Record.getMatrixHeaderField(options)`, `Record.getMatrixHeaderValue(options)`, and `Record.getMatrixHeaderCount(options)`.

When on an existing line in the matrix, use `Record.getCurrentMatrixSublistValue(options)` and `Record.setCurrentMatrixSublistValue(options)` to get and set the price on that line in the specific column, respectively.

For all other lines in the matrix, use `Record.getMatrixSublistValue(options)` and `Record.findMatrixSublistLineWithValue(options)`.

**Standard Sublist APIs**

If you want to reference the other fields in the Pricing sublist, such as currency, name, or discount, use the existing `Record.getSublistValue(options)` and pass in the existing fieldId (for example: `price1currency`). Also see Pricing Sublist Field IDs, which specifies which fields on the Pricing sublist can be set using standard sublist APIs (within the `N/record Module` and `N/currentRecord Module` modules).

**Example:**

```javascript
// load an item record
var record = record.load({
    type: record.Type.INVENTORY_ITEM,
    id: 536          // for example - use your actual record id
});

// get the value of the currency field on line 2
var currency = record.getSublistValue({
    sublistId: 'price1',
    fieldId: 'currency',
    line: '2'
});

// get the value of the pricelevelname field on line 2
var pricelevelname2 = record.getSublistValue({
    sublistId: 'price1',
    fieldId: 'pricelevel',
    line: '2'
});

var pricelevelname3 = record.getSublistValue({
```
Records that Include the Pricing Sublist

The Pricing sublist appears on the following records that are scriptable using SuiteScript: Assembly Item, Lot Numbered Assembly Item, Serialized Assembly Item, Lot Numbered Inventory Item, Serialized Inventory Item, Gift Certificate Item, Kit Item, Inventory Item.

Assembly Item

Note: This topic applies to all versions of SuiteScript.

An assembly item is an inventory item made of several components, but identified as a single item. Assemblies are manufactured by combining raw materials you stock.

You create assembly item records to define the members of an assembly, then NetSuite enables you to track both the raw materials and the assembled items separately.

For example, an Wolfe Electronics sells a computer called Creativo 2400 that they assemble in-house. The Creativo 2400 computer is assembled from these inventory components: one Superion 2 GHz processor, one Creativo 2400 motherboard, 1 GB RAM, 80 GB Hard Drive, one Superion sound card, and one power supply.

NetSuite tracks the stock of the Creativo 2400 and each component item separately. Then, Wolfe can track the stock level of Creativo 2400 in inventory and available to ship to customers, and the quantity of materials available to assemble more.

For help working with this record in the UI, see the help topic Assembly Items.
The internal ID for this record is **assemblyitem**. This record is also sometimes referred to as Build/Assembly.

See the [SuiteScript Records Browser](https://www.netsuite.com/help/) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](https://www.netsuite.com/help/) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The assembly item record is scriptable in server SuiteScript only.

**Supported Functions**

The assembly item record is fully scriptable — it can be created, updated, copied, deleted, and searched using server SuiteScript. It can also be transformed.

**Usage Notes**

The taxschedule field of the Members sublist is visible only in the UI when the Advanced Taxes feature is enabled.

The assembly item record includes the Pricing Sublist / Pricing Matrix.

**Code Samples**

The following sample changes the item name of an assembly.

```javascript
var assembly = record.load({
  type: record.Type.ASSEMBLY_ITEM,
  id: 123
});
assembly.setValue({
  fieldId: 'itemid',
  value: 'new name'
});
var recordId = assembly.save();
```

**Description Item**

**Note:** This topic applies to all versions of SuiteScript.

Description line items let you put sentence- or paragraph-long descriptions on items you are not actually selling. For example, you may want to enter special shipping instructions or a disclaimer.
Description items have no amount field. They are only used to add text to transactions. They can be used on both purchase and sales transactions. The item name of a description item does not appear on printed forms, only the descriptive text. Nothing appears in the amount column for description items.

As you create a transaction, select a description item from the items list. For help working with this record in the UI, see the help topic Description Items. The internal ID for this record is descriptionitem. See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

## Supported Script Types

The description record is scriptable in both client and server SuiteScript.

## Supported Functions

The description record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

### Discount Item

**Note:** This topic applies to all versions of SuiteScript.

Discount item records are used to create discounts you can apply to your transactions. For help working with this record in the UI, see the help topic Discount Items.

The internal ID for this record is discountitem. See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
Supported Script Types

The discount record is scriptable in both client and server SuiteScript.

Supported Functions

The description record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Download Item

Note: This topic applies to all versions of SuiteScript.

You create download item records for files that you want customers to be able to purchase and download in your Web store.

Customers are charged per download item as opposed to per item. For example, if you want to charge customers for music downloads per song, you would create an item for each song. If you want to charge customers per album, you would create one item and attach each song for the album.

If a download requires a license code, such as a software download, you can add the license code on the Downloads subtab of the customer's record. When a license code is added, the code is available to the customer in the Customer Center or the My Account tab of your website and included in invoice email notification.

For help working with this record in the UI, see the help topic Download Items.

The internal ID for this record is downloaditem.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The download item record is scriptable in both client and server SuiteScript.

Supported Functions

The download item record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Gift Certificate Item

**Note:** This topic applies to all versions of SuiteScript.

You can create gift certificate items that allow customers to purchase store credit they can send to someone as a gift. The recipient uses the gift certificate code when placing an order through your Web store or entering a transaction with a sales representative. You can set a preference in NetSuite for how you want to generate the gift certificate codes: you can create them yourself, or use a random hash code automatically generated by the system.

Gift certificate codes are not active until the order used to purchase the gift certificate is billed.

For help working with this record in the UI, see the help topic Gift Certificates.

The internal ID for this record is `giftcertificateitem`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The gift certificate record is scriptable in both client and server SuiteScript.

**Supported Functions**

The gift certificate item record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

The gift certificate item record includes the Pricing Sublist / Pricing Matrix.

Inventory Item

**Note:** This topic applies to all versions of SuiteScript.

NetSuite inventory item records enable you to track the quantity and value of your inventory. Your balance sheet will automatically reflect the value of your inventory on hand, and your income statement will automatically reflect the markup you charge for these items.

This record is also referred to as Inventory Part in the UI.
For help working with this record in the UI, see the help topic Inventory Items.

The internal ID for this record is inventoryitem.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The inventory item record is scriptable in both client and server SuiteScript.

**Supported Functions**

The inventory item record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

The inventory item record includes the Pricing Sublist / Pricing Matrix.

**Item Group**

Note: This topic applies to all versions of SuiteScript.

An item group is stocked and sold as a single unit, but may actually be made up of several individual items.

For help working with this record in the UI, see the help topic Item Groups.

The internal ID for this record is itemgroup.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
Supported Script Types

The item group record is scriptable in both client and server SuiteScript.

Supported Functions

The item group record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Item Search

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see . The internal ID for this record is item.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The item search record is a search record only.

Supported Functions

The item search record is a search record only. You cannot create, update, delete, or copy this record.

Kit Item

Note: This topic applies to all versions of SuiteScript.

Kits or packages let you create items that are collected from other items. You can assign multiple price levels to your kits and even make them available in your website. Whenever you sell a kit, inventory items are deducted from inventory.

For help working with this record in the UI, see the help topic Kit/Package Items.

The internal ID for this record is kititem.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The kit record is scriptable in server SuiteScript.

Supported Functions

The kit record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

The taxschedule field of the Members sublist is visible only in the UI when the Advanced Taxes feature is enabled.

The kit item record includes the Pricing Sublist / Pricing Matrix.

Lot Numbered Assembly Item

Note: This topic applies to all versions of SuiteScript.

Lot numbered assembly items enable you to build items from raw materials and track the inventory of both the finished items and the raw materials separately. The completed assembly is assigned a lot number to track it as it enters and leaves your inventory.

For help working with this record in the UI, see the help topic Lot Numbered Items. For help working with assembly items in the UI information, see the help topic Assembly Items.

The internal ID for this record is lotnumberedassemblyitem.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types

The lot numbered assembly item record is scriptable in both client and server SuiteScript.

Supported Functions

The lot numbered assembly item record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

The taxschedule field of the Members sublist is visible only in the UI when the Advanced Taxes feature is enabled.

The lot numbered assembly item record includes the Pricing Sublist / Pricing Matrix.

Lot Numbered Inventory Item

Note: This topic applies to all versions of SuiteScript.

Lot numbered inventory items track the purchase, stock, and sale of groups of items by assigning lot numbers. Lot numbered item records track the quantity of items and the specific cost for each lot as products are purchased and sold.

For help working with this record in the UI, see the help topic Lot Numbered Items. For help about using inventory items in the UI, see the help topic Inventory Items.

The internal ID for this record is lotnumberedinventoryitem.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The lot numbered inventory item record is scriptable in both client and server SuiteScript.

Supported Functions

The lot numbered inventory item record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Usage Notes

The lot numbered inventory item record includes the Pricing Sublist / Pricing Matrix.

Markup Item

Note: This topic applies to all versions of SuiteScript.

You can use markup items to apply an additional charge to an order. Using markup items enables you to track markup amounts without affecting inventory valuation.

For example, you may want to charge a rush fee for completing a service or delivering an item quicker than is usually guaranteed. You can choose to markup the amount for this charge by a flat additional fee.

For help working with this record in the UI, see the help topic Markup Items.

The internal ID for this record is markupitem.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The markup record is scriptable in both client and server SuiteScript.

Supported Functions

The markup record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Non-Inventory Part

Note: This topic applies to all versions of SuiteScript.

Items that you always drop ship or other items that you sell or purchase but do not stock can be recorded and tracked as non-inventory items.

For help working with this record in the UI, see the help topic Non-Inventory Items.

The internal ID for this record is noninventoryitem.
See the **SuiteScript Records Browser** for all internal IDs associated with this record.

![Note:](#) For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- **SuiteScript 2.0 Scripting Records and Subrecords**
- **SuiteScript 2.0 Custom Forms**
- **N/record Module**

**Supported Script Types**

The non-inventory record is scriptable in both client and server SuiteScript.

**Supported Functions**

The non-inventory part record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Other Charge Item**

![Note:](#) This topic applies to all versions of SuiteScript.

Other charge items can be used to designate items or services you purchase or sell that do not fall into another type of item.

For help working with this record in the UI, see the help topic **Other Charge Items**.

The internal ID for this record is `otherchargeitem`.

See the **SuiteScript Records Browser** for all internal IDs associated with this record.

![Note:](#) For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- **SuiteScript 2.0 Scripting Records and Subrecords**
- **SuiteScript 2.0 Custom Forms**
- **N/record Module**

**Supported Functions**

The other charge item record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Payment Item

**Note:** This topic applies to all versions of SuiteScript.

You can create payment items for types of payments that are made to invoices and should show separately. For example, you may want to create a payment item to specify a down payment amount.

For help working with this record in the UI, see the help topic Payment Items.

The internal ID for this record is `paymentitem`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

## Supported Script Types

The payment record is scriptable in both client and server SuiteScript.

## Supported Functions

The payment record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

## Reallocate Items

**Note:** This topic applies to all versions of SuiteScript.

NetSuite sometimes automatically allocates items from inventory to commit them to fill orders. Inventory may be automatically committed in the following cases:

- As each sales order is created or approved, your account automatically allocates inventory from the item's quantity available.
- Inventory is automatically committed to fill backorders when goods are received from your vendors.
- The inventory generated from assembly work orders is automatically allocated.

For help working with this record in the UI, see the help topic Reallocating Items.

The internal ID for this record is `reallocateitem`.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Items

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The reallocate items record is only exposed in user event scripts. You can execute `beforeLoad`, `beforeSubmit`, and `afterSubmit` user event scripts on this record.

Supported Functions

The reallocate items record can only be copied and searched. It cannot be created, updated, or deleted using scripts.

Serialized Assembly Item

Note: This topic applies to all versions of SuiteScript.

Serialized assembly items enable you to build items from raw materials and track the inventory of both finished items and the raw materials separately. The completed assembly is assigned a serial number to track it as it enters and leaves your inventory. Serialized assembly items are available on sales transactions and inventory adjustment transactions. They are not available on purchase transactions.

For help working with this record in the UI, see the help topic Serial Numbered Items. For help about using assembly items in the UI, see the help topic Assembly Items.

The internal ID for this record is `serializedassemblyitem`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The serialized assembly item record is scriptable in both client and server SuiteScript.
Supported Functions

The serialized assembly record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

The serialized assembly item record includes the Pricing Sublist / Pricing Matrix.

Serialized Inventory Item

Note: This topic applies to all versions of SuiteScript.

Serialized inventory item records are used to track information about items which you maintain a stock of. Note that you must first enable serialized inventory items in your NetSuite account before you can access this record type. To enable serialized inventory items, go to Setup > Company > Enable Features. On the Items & Inventory tab, under Inventory, select the Serialized Inventory box.

For help working with this record in the UI, see the help topic Serial Numbered Items. For help about using inventory items in the UI, see the help topic Inventory Items.

The internal ID for this record is serializedinventoryitem.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The serialized inventory item record is scriptable in both client and server SuiteScript.

Supported Functions

The serialized inventory record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

The serialized inventory item record includes the Pricing Sublist / Pricing Matrix.
Service

Note: This topic applies to all versions of SuiteScript.

A service item is an item you create to track time and record billable hours.
For help working with this record in the UI, see the help topic Service Items.
The internal ID for this record is serviceitem.
See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The service record is scriptable in both client and server SuiteScript.

Supported Functions

The service record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Shipping Item

Note: This topic applies to all versions of SuiteScript.

A shipping item is a delivery method for a shipping carrier. It describes how to ship an item and can include shipping rate information, handling rates, and rules for shipping and handling. It can also specify when shipping is free.
For help working with this record in the UI, see the help topic Shipping Items.
The internal ID for this record is shipitem.
See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
- SuiteScript 2.0 Scripting Records and Subrecords
Supported Script Types

The shipping item record is supported in all server-side scripts. Client-side scripting is not supported.

Supported Functions

The shipping item record can only be updated, deleted, and searched. It cannot be created or copied. You can create this record in the UI, however.

Note: Create and copy are not supported. You can create this record only in the UI.

Usage Notes

The following shipping item record components are not scriptable:

- Handling Table
- Shipping Table
- Free Shipping Items Tab

Subscription Plan

Note: This topic applies to all versions of SuiteScript.

The subscription plan record is used to build information for subscriptions.

The subscription plan record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features (Administrator), on the Transactions subtab. When the feature is enabled, you can access a list of subscription plan records in the UI by going to Lists > Subscriptions > Subscription Plans.

For help working with this record in the UI, see the help topic Creating Subscription Plans.

The internal ID for this record is subscriptionplan.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types

The subscription plan record is scriptable in server and client SuiteScript.

Supported Functions

The subscription plan record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Subtotal Item

Note: This topic applies to all versions of SuiteScript.

Subtotal item records are used to separate groups or individual line items if you want to offer a discount or tax on certain line items but not others.

For help working with this record in the UI, see the help topic Subtotal Items.

The internal ID for this record is subtotalitem.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The subtotal record is scriptable in both client and server SuiteScript.

Supported Functions

The subtotal record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Supply Chain Snapshot Simulation

Note: This topic applies to all versions of SuiteScript.

The supply chain snapshot simulation record is used to provide a simulation of a transaction to determine what the effect that transaction would have on your inventory.
The supply chain snapshot simulation record is only available when the Supply Chain Control Tower feature is enabled at Setup > Company > Enable Features, on the Items & Inventory subtab. When the feature is enabled, you can access a list of supply chain snapshot simulation records in the UI by going to Lists > Supply Chain > Supply Chain Snapshot Simulations.

For help working with the supply chain snapshot record in the UI, see the help topic Supply Chain Control Tower.

The internal ID for this record is supplychainsnapshotsimulation.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The supply chain snapshot simulation record is scriptable in server and client SuiteScript.

**Supported Functions**

The supply chain snapshot simulation record can be created, updated, deleted, and searched using SuiteScript.

**Usage Notes**

Item type is not scriptable.

The system notes are search only.

**Code Samples**

The following sample shows how to create a supply chain snapshot simulation.

```javascript
function createSimulation(location, item, quantity, date) {
    var simulationRecord = nlapiCreateRecord('supplychainsnapshotsimulation', {recordmode: 'dynamic'});
    simulationRecord.setFieldValue('memo', 'Simulation for item id=' + item);
    simulationRecord.setFieldValue('transactiontype', 'PurchOrd');
    simulationRecord.setFieldValue('quantity', quantity);
    simulationRecord.setFieldValue('status', '5');
    simulationRecord.setFieldValue('itemnamenumber', item);
    simulationRecord.setFieldValue('supplylocationdate', date);
    simulationRecord.setFieldValue('supplylocationsubsidiary', 1);
}
```
Communications

Note: This topic applies to all versions of SuiteScript.

The following communication records are scriptable in SuiteScript:

- Message
- Note

Message

Note: This topic applies to all versions of SuiteScript.

A message is used to record correspondence you have with a specific business. Use the message record to add an email message to an existing customer, contact, or opportunity record. After an email message has been added to a record, any related emails are automatically attached to the same record as well as to any recipients of the original email.

For help working with this record in the user interface, see the help topic Sending Email from Records.

The internal ID for this record is message.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The message record is scriptable in server SuiteScript only.

Supported Functions

For new messages, full scripting is supported. This support enables you to do the equivalent of clicking the Attach button in the UI for email messages. Note that there are also other types of messages, but those are not supported. In the message that you are creating, you can set all supported fields for email
messages before saving. After you save the new message, there are restrictions applied to it. You cannot edit the existing message. It is possible to delete an existing message, if the permissions for the user are set to allow this type of operation. However, this operation is not recommended, because it can cause email communication threads to be displayed incorrectly.

Usage Notes

- Message records can be edited only during the create operation. After they are created and submitted, existing message records cannot be edited.
- Existing message records can be copied and deleted.
- Only beforeLoad and afterSubmit user event scripts will execute on the Message record type when a message is created by an inbound email case capture. Scripts set to execute on a beforeSubmit event will not execute.

For example, if you have a test script like the following deployed to the Message record type:

```javascript
function beforeLoad(type, name) {
    log.debug ( {
        title: 'Before Load',
        details: 'In beforeLoad function'
    });
}

function beforeSubmit(type, name) {
    log.debug ( {
        title: 'Before Submit',
        details: 'In beforeSubmit function'
    });
}

function afterSubmit(type, name) {
    log.debug ( {
        title: 'After Submit',
        details: 'In afterSubmit function'
    });
}
```

only the beforeLoad(...) and afterSubmit(...) functions will execute if the message was created to respond to an inbound emailed case.

- When creating a new message, you can use the ccbccclist sublist to add an email address to the cc or bcc fields. For an example, see Code Sample.

- NetSuite 2016.2 introduced a new requirement for creating a message. An entity that is defined as the author or recipient for a newly created message must have an associated email in NetSuite. If the NetSuite record for an entity defined as the author of a message does not include an email, the authoremail field must be defined on the message record. If the NetSuite record for an entity defined as the recipient of a message does not include an email, the recipientemail field must be defined on the message record. Prior to 2016.2, the authoremail and recipientemail fields were not mandatory for the message record.

Code Sample

The following SuiteScript 2.0 snippet shows how to add an email address to the cc line of a new message, when working in standard (deferredDynamic) mode.

```javascript
...```
```javascript
var messageRec = record.create({
    type: record.Type.MESSAGE,
});
messageRec.insertLine({
    sublistId: 'ccbcclist',
    line: 0
});
messageRec.setSublistValue({
    sublistId: 'ccbcclist',
    fieldId: 'email',
    value: 'john@smith.com',
    line: 0
});
messageRec.setSublistValue({
    sublistId: 'ccbcclist',
    fieldId: 'cc',
    value: 'true',
    line: 0
});
```

**Note**

This topic applies to all versions of SuiteScript.

Notes are used to attach information to another record. Use the notes record to create new notes and attach them to a specific record.

The internal ID for this record is `note`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

**Supported Script Types**

The note record is scriptable in server SuiteScript only.

**Supported Functions**

The note record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Transactions

**Note:** This topic applies to all versions of SuiteScript.

A macro can be used for taxable transactions. For more information, see Transaction Record Macros.

The following transaction records are scriptable in SuiteScript:

- Advanced Intercompany Journal Entry
- Assembly Build
- Assembly Unbuild
- Balance Transactions by Segments
- Bin Putaway Worksheet
- Bin Transfer
- Blanket Purchase Order
- Cash Refund
- Cash Sale
- Change Order
- Charge
- Check
- Credit Card Charge
- Credit Card Refund
- Credit Memo
- Custom Transaction
- Customer Deposit
- Customer Payment
- Customer Refund
- Deposit
- Deposit Application
- Estimate
- Expense Report
- Fulfillment Request
- GL Audit Numbering Sequence
- Intercompany Journal Entry
- Intercompany Transfer Order
- Inventory Adjustment
- Inventory Cost Revaluation
- Inventory Count
- Inventory Detail
- Inventory Status Change
- Inventory Transfer
- Invoice
- Item Demand Plan
- Item Fulfillment
- Item Receipt
- Item Supply Plan
- Journal Entry
- Landed Cost
- Manufacturing Operation Task
- Manufacturing Planned Time
- Memorized Transaction Definition
- Multi-Book Accounting Transaction
- Opportunity
- Order Schedule
- Paycheck
- Paycheck Journal
- Period End Journal
- Purchase Contract
- Purchase Order
- Requisition
- Return Authorization
- Revenue Arrangement
- Revenue Commitment
- Revenue Commitment Reversal
- Sales Order
- Statistical Journal Entry
- Store Pickup Fulfillment
- Subscription
- Subscription Line
- Time
- Transaction Search
- Transfer Order
- Unlocked Time Period
- Usage
- Vendor Bill
- Vendor Credit
- Vendor Payment
- Vendor Prepayment
- Vendor Prepayment Application
- Vendor Return Authorization
- Weekly Timesheet
- Work Order
- Work Order Close
- Work Order Completion
Transaction Record Macros

The `calculateTax` and `getSummaryTaxTotals` macros can only be used for the following taxable transactions, if the SuiteTax feature is enabled. If a macro is executed on a non-taxable transaction or the SuiteTax feature is not enabled, an error occurs.

- Cash Refund
- Cash Sale
- Credit Card Charge
- Credit Memo
- Estimate
- Expense Report
- Invoice
- Opportunity
- Purchase Order
- Return Authorization
- Sales Order
- Vendor Bill
- Vendor Credits
- Vendor Return Authorization

### calculateTax

<table>
<thead>
<tr>
<th>Macro Description</th>
<th>Calculates taxes on a transaction, just as when a user clicks the Preview Tax button in the UI. Taxes are always calculated when the transaction is saved.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client and server scripts  
For additional information, see the help topic SuiteScript 2.0 Script Types. |
| Since             | 2016.1                                                                                           |

### Parameters

See the help topic `Record.executeMacro(options)` for details about parameters required for the execution of any macro.

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_UNSUPPORTED_METHOD</td>
<td>The SuiteTax feature is disabled, or the transaction type of the current record is non-taxable.</td>
</tr>
<tr>
<td>SSS_TAX_REGISTRATION_REQUIRED</td>
<td>The subsidiary of the current record does not have a valid tax registration.</td>
</tr>
</tbody>
</table>
Transaction calculateTax Macro Syntax

```javascript
require(['N/currentRecord'], function(currentRecord){
    currentRecord.executeMacro({id: 'calculateTax'});
});
```

For an example of how to call the calculateTax macro on a sales order record, see Record Macro Script Samples.

getSummaryTaxTotals

<table>
<thead>
<tr>
<th>Macro Description</th>
<th>Checks whether tax is calculated, iterates through tax details, and gathers all tax type IDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>An object that contains notification and response, which contains the <code>taxTotals</code> array</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server scripts</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**Parameters**

See the help topic `Record.executeMacro(options)` for details about parameters required for the execution of any macro.

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_UNSUPPORTED_METHOD</td>
<td>The SuiteTax feature is disabled, or the transaction type of the current record is non-taxable.</td>
</tr>
<tr>
<td>SSS_TAX_REGISTRATION_REQUIRED</td>
<td>The subsidiary of the current record does not have a valid tax registration.</td>
</tr>
</tbody>
</table>

Transaction getSummaryTaxTotals Macro Syntax

```javascript
require(['N/currentRecord'], function(currentRecord){
    var taxTotals = currentRecord.executeMacro({id: 'getSummaryTaxTotals'});
});
```

Sample Return Objects

```json
{
    "taxTotals": [
        {
            "taxTypeId": 123,
            "taxTypeName": 'VAT Standard',
            "taxTotal": '833.33'
        }
    ]
}
```
Advanced Intercompany Journal Entry

**Note:** This topic applies to all versions of SuiteScript.

An advanced intercompany journal entry includes all the operations of the original journal entries, in addition to new time-saving functions. For SuiteScript details on the original intercompany journals, see Intercompany Journal Entry.

In the UI, you can access this record at Transactions > Financial > Make Advanced Intercompany Journal Entries.

If your account has the Multi-Book Accounting feature enabled, you can also work with book-specific intercompany journal entry records, which are available at Transactions > Financial > Make Book Specific Advanced Intercompany Journal Entries. Although they have different entry forms, both book-specific and regular intercompany journal entries are the same record type. Within SuiteScript, they are differentiated by the accountingbook field. In other words, a record that has a value set for accountingbook is book-specific. Otherwise, the record is not a book-specific intercompany journal entry.

For help working with this record in the UI, see the help topics Making Advanced Intercompany Journal Entries and Book-Specific Advanced Intercompany Journal Entries.

The internal ID for this record is advintercompanyjournalentry.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The intercompany journal entry record is scriptable in both client and server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

### Supported Functions

The advanced intercompany journal entry record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
**Warning:** If you update any type of journal entry that has been applied as a payment to an invoice or vendor bill, the relationship between the journal entry and payment is removed and the payment is no longer applied.

## Code Samples

The following sample shows how to create an advanced intercompany journal entry.

```javascript
var journalEntry = record.create(
    type: record.Type.ADV_INTER_COMPANY_JOURNAL_ENTRY,
    isDynamic: true
);
journalEntry.setValue(
    fieldId: 'subsidiary',
    value: 1
);
journalEntry.selectNewLine(
    sublistId: 'line'
);
journalEntry.setCurrentSublistValue(
    sublistId: 'line',
    fieldId: 'subsidiary',
    value: 1
);
journalEntry.setCurrentSublistValue(
    sublistId: 'line',
    fieldId: 'account',
    value: 100
);
journalEntry.setCurrentSublistValue(
    sublistId: 'line',
    fieldId: 'debit',
    value: '1.00'
);
journalEntry.commitLine(
    sublistId: 'line'
);

journalEntry.selectNewLine(
    sublistId: 'line'
);
journalEntry.setCurrentSublistValue(
    sublistId: 'line',
    fieldId: 'subsidiary',
    value: 1
);
journalEntry.setCurrentSublistValue(
    sublistId: 'line',
    fieldId: 'account',
    value: '101'
);
journalEntry.setCurrentSublistValue(
    sublistId: 'line',
    fieldId: 'credit',
    value: '1.00'
);
```
```javascript
  value: '1.00'
});
journalEntry.commitLine({
    sublistId: 'line'
});

journalEntry.selectNewLine({
    sublistId: 'line'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'subsidiary',
    value: 2
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: '200'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'credit',
    value: '1.00'
});
journalEntry.commitLine({
    sublistId: 'line'
});

journalEntry.selectNewLine({
    sublistId: 'line'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'subsidiary',
    value: 2
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: '201'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'debit',
    value: '1.00'
});
journalEntry.commitLine({
    sublistId: 'line'
});
```
Assembly Build

**Note:** This topic applies to all versions of SuiteScript.

An assembly item is an inventory item made of several components, but identified as a single item. This type of item lets you define the members of an assembly and to separately track both the component items and the assembled items in inventory. For each assembly build, the assembly item stock level increases and the member items' individual stock levels decrease.

An assembly build transaction records the physical manufacture of an assembly item from component items and the related inventory level changes.

For details about this type of transaction, see [Building Assembly Items](#). For details about this type of item, see the help topic [Assembly Items](#).

The internal ID for this record is `assemblybuild`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

Also note that this record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

**Supported Script Types**

The assembly build record is scriptable in server SuiteScript only.

**Supported Functions**

The assembly build entry record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Assembly Unbuild

**Note:** This topic applies to all versions of SuiteScript.

An assembly item is an inventory item made of several components, but identified as a single item. This type of item lets you define the members of an assembly and to separately track both the component items and the assembled items in inventory.

An assembly unbuild transaction records the physical taking apart of an assembly item back into its component items and the related inventory level changes. For each assembly unbuild, the assembly item stock level decreases and the member items' individual stock levels increase.
For details about this type of transaction, see Building Assembly Items. For details about this type of item, see the help topic Assembly Items.

For help working with this record in the UI, see the help topic Unbuilding Assembly Items.

The internal ID for this record is assemblyunbuild.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The assembly unbuild record is scriptable in server SuiteScript only.

**Supported Functions**

The assembly unbuild entry record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Balance Transactions by Segments**

Note: This topic applies to all versions of SuiteScript.

The balance transactions by segments record lets you select the criteria for transactions to include in the balancing by segments process. This process generates one or more balancing journals to balance the debits and credits of the included transactions by balancing segment.

This record and process are part of the Balancing Segments feature. For information, see the help topic Balancing Segments and Journals. The Custom Segments feature is a prerequisite for the Balancing Segments feature.

Before you begin working with this record programmatically, see the help topic Setup for Balancing Segments.

For help working with this record in the UI, see the help topic Balancing Transactions by Segment.

The internal ID for this record is balancetrxbys segments.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
Supported Script Types

The balance transactions by segments record is scriptable in server SuiteScript only.

Supported Functions

The balance transactions by segments record is partially scriptable. You can create, read, and search it using SuiteScript. You cannot update, copy, transform, or delete it.

Usage Notes

The Transactions and Results subtabs, visible in the UI in view mode, are not scriptable.

Code Sample

The following sample shows how to run the balancing by segments process in a NetSuite OneWorld account with the Multi-Book Accounting feature enabled.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/record'], function(record) {
  var balancingTrxBySegments = record.create({
    type: record.Type.BALANCE_TRX_BY_SEGMENTS,
    isDynamic: true
  });
  balancingTrxBySegments.setValue({
    fieldId: 'accountingbook',
    value: [1, 2]
  });
  balancingTrxBySegments.setValue({
    fieldId: 'subsidiary',
    value: [3, 4]
  });
  balancingTrxBySegments.setValue({
    fieldId: 'includechildren',
    value: true
  });
  balancingTrxBySegments.setValue({
    fieldId: 'postingperiod',
    value: 64
  });
  balancingTrxBySegments.setValue({
    fieldId: 'memo',
    value: 'My Balancing'
  });
  balancingTrxBySegments.save();
});
```
Bin Putaway Worksheet

**Note:** This topic applies to all versions of SuiteScript.

Use the Bin Putaway Worksheet to print a list of bin numbers for items that need to be re-stocked in your warehouse or stock room.

Items are added to this list any time an item that uses bins has a greater total on-hand count than the combined on-hand count of its associated bins.

How the worksheet looks and functions depends on whether you use basic Bin Management or use Advanced Bin / Numbered Inventory Management.

For help working with this record in the UI, see the help topics Basic Bin Putaway Worksheet and Advanced Bin Putaway Worksheet.

The internal ID for this record is `binworksheet`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The bin putaway worksheet record can only be scripted in dynamic mode. For details about dynamic scripting, see the following help topics:

- SuiteScript 1.0 Working with Records in Dynamic Mode
- SuiteScript 1.0 How do I enable dynamic mode?
- SuiteScript 1.0 Is dynamic mode better than standard mode?
- SuiteScript 1.0 Standard vs. Dynamic Mode Code Samples

Note that client (remote object) scripting does not support dynamic scripting.

**Supported Functions**

The bin putaway worksheet record is partially scriptable — it can be created, deleted, and searched using SuiteScript. It cannot be copied or updated.

**Bin Transfer**

**Note:** This topic applies to all versions of SuiteScript.

You can record a bin transfer to move items between bins within a warehouse.
For example, at your East Coast location, you stock item #AB123 in two bins: bin #3003 and #4004. There are currently 100 of the item in each bin. You can enter a bin transfer to record the transfer of 50 items out of bin #3003 and into bin #4004.

Recording a bin transfer does not post to your chart of accounts and has no financial impact. The transfer only updates the quantity on hand in each bin for the items transferred.

On the bin transfer record, identify the item, the bin the item will come from, the bin the item will move into and the quantity to be moved.

For help working with this record in the UI, see the help topic Bin Transfers.

The internal ID for this record is bintransfer.

The bin transfer record contains a subrecord: Inventory Detail. For details about subrecords in SuiteScript, see the help topic SuiteScript 2.0 Scripting Subrecords.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The bin transfer is scriptable in server SuiteScript only.

Supported Functions

The bin transfer record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Blanket Purchase Order

Note: This topic applies to all versions of SuiteScript.

With this record, you can take advantage of fixed pricing for a preset number of items that you will buy during a specific time period. This approach lets you avoid sporadic pricing negotiations with vendors.

This record is available only when the Blanket Purchase Order feature is enabled at Setup > Company > Enable Features, on the Transactions subtab.

In the UI, you access this record at Transactions > Purchases > Enter Blanket Purchase Order. For help working with this record in the UI, see the help topic Creating a Blanket Purchase Order.

The internal ID of this record is blanketpurchaseorder.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Transactions

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The blanket purchase order record is scriptable in both client and server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

The blanket purchase order record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Cash Refund**

**Note:** This topic applies to all versions of SuiteScript.

A cash refund transaction records the return of money to a customer who immediately paid for goods or services using cash, a check or a credit card.

For help working with this record in the UI, see the help topic Refunding a Cash Sale.

The internal ID for this record is `cashrefund`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The cash refund record is scriptable in both client and server SuiteScript.

**Supported Functions**

The cash refund record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Usage Notes

See the following sections for more details on working with this record:

- **Fields**
- **Email Operations**

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

## Fields

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccnumber</td>
<td>Credit Card #</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

## Email Operations

NetSuite enables a user to email a copy of a transaction to a customer or another recipient.

If you deploy a beforeLoad user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction. However, if the companywide preference **Use Popup for Main Transaction Email Button** is enabled, a beforeLoad script will not execute in certain cases, depending on how the email is sent. That is, the script will not execute if the user views the transaction and selects Actions > Email, or edits the transaction and sends the email by using the Communications tab. In these cases, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this window are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the **Message** record type.

An additional way of emailing a transaction is by editing the transaction and using the Save & Email button. Note that, even if the Use Popup for Main Transaction Email Button preference is enabled, the popup window is not available for emails sent this way. For that reason, a beforeLoad user event script deployed on the transaction will always be triggered by use of the Save & Email button.

For more information on the Use Popup for Main Transaction Email Button preference, see the help topic **Setting Email Preferences**. For details about emailing transactions, see the help topic **Emailing Transactions**.
Cash Sale

**Note:** This topic applies to all versions of SuiteScript.

A cash sale is a transaction that records the sale of goods or services for which you receive immediate payment. Enter a cash sale when payment for goods or services has been received at the time of delivery.

Record cash sales by adding each item sold from your Items list. Then, cash sale line-items specify the goods and services sold and their sales amounts. The sum of all sales amounts plus any applicable tax equals the total amount paid for this sale.

For help working with this record in the UI, see the help topic **Cash Sales**.

The internal ID for this record is `cashsale`.

See the [SuiteScript Records Browser](https://www.netsuite.com) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic **Working with the SuiteScript Records Browser** in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](https://www.netsuite.com)
- [SuiteScript 2.0 Custom Forms](https://www.netsuite.com)
- [N/record Module](https://www.netsuite.com)

**Supported Script Types**

The cash sale record is scriptable in both client and server SuiteScript.

**Supported Functions**

The cash sale record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

See the following sections for more details on working with this record:

- **Fields**
- **Email Operations**

The following cash sale record sublists are **list** sublists:

- Billable Expenses
- Billable Items
- Billable Time

In the NetSuite Help Center, see the help topic **List Sublists** for information on this sublist type.
The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

The Shipping sublist's Shipping Tax Code field, internal ID shippingtaxcode, appears only if per-line taxes have been set on the Item sublist.

## Fields

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccnumber</td>
<td>Credit Card #</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>isrecurringpayment</td>
<td>Recurring Payment</td>
<td>A value for this field is stored only if the value for paymentmethod is a credit card.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

Also note that this record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

## Email Operations

NetSuite enables a user to email a copy of a transaction to a customer or another recipient.

If you deploy a beforeLoad user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction. However, if the companywide preference **Use Popup for Main Transaction Email Button** is enabled, a beforeLoad script will not execute in certain cases, depending on how the email is sent. That is, the script will not execute if the user views the transaction and selects Actions > Email, or edits the transaction and sends the email by using the Communications tab. In these cases, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this window are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the Message record type.

An additional way of emailing a transaction is by editing the transaction and using the Save & Email button. Note that, even if the Use Popup for Main Transaction Email Button preference is enabled, the popup window is not available for emails sent this way. For that reason, a beforeLoad user event script deployed on the transaction will always be triggered by use of the Save & Email button.
For more information on the Use Popup for Main Transaction Email Button preference, see the help topic Setting Email Preferences. For details about emailing transactions, see the help topic Emailing Transactions.

Change Order

Note: This topic applies to all versions of SuiteScript.

The change order record is used to transact changes to subscriptions. For example, a change order is required to activate a drafted subscription.

The change order record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features (Administrator), on the Transactions subtab. When the feature is enabled, you can access a list of change order records in the UI by going to Transactions > Subscriptions > View Subscription Change Orders.

For help working with this record in the UI, see the help topic Condition for Prospective Merge Creation.

The internal ID for this record is subscriptionchangeorder.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- Condition for Prospective Merge Creation
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The change order record is scriptable in server and client SuiteScript.

Supported Functions

The change order record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Charge

Note: This topic applies to all versions of SuiteScript.

The charge record is used to represent a single billable amount that a client must pay.

The charge record is available only when the Charge-Based Billing feature is enabled at Setup > Company > Enable Features, on the Transactions subtab. When the feature is enabled, you can access the charge record in the UI by choosing Transactions > Customers > Create Charges > List.
For help working with this record in the UI, see the help topic Generating Charges.

The internal ID for this record is charge.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The charge record is scriptable in server SuiteScript only.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

The charge record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Code Samples**

The following sample shows how to create a charge record.

```javascript
var charge = record.create({
   type: record.Type.CHARGE
});
charge.setText({
   fieldId: 'stage',
   text: 'Ready'
});
charge.setValue({
   fieldId: 'chargeType',
   value: 'Time'
});
charge.setValue({
   fieldId: 'billto',
   value: '43'
});
charge.setValue({
   fieldId: 'chargedate',
   value: '6/10/2012'
});
charge.setValue({
   fieldId: 'salesorder',
   value: '122'
});
charge.setValue({
```

SuiteScript Records Guide
Check

Note: This topic applies to all versions of SuiteScript.

A check transaction creates and records a check used to pay an expense, records an expense paid in cash and not entered as a bill, or records a non-check debit transaction, such as a debit card transaction, ATM (automated teller machine) transaction, or EFT (electronic funds transfer) payment. A check transaction records an expense directly to your books by debiting the expense account specified in the transaction detail and crediting the bank account the check selected for the check.

For help working with this record in the UI, see the help topic Checking.

The internal ID for this record is check.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types

The check record is scriptable in both client and server SuiteScript.

Supported Functions

The check record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

The following sections provide usage notes for specific fields and sublists.

Using Landed Cost Fields

When you create a landed cost category, the associated field IDs for the first category are landedcostamount1 and landedcostsource1. If you create a second category, the IDs will be landedcostamount2 and landedcostsource2.

This pattern increments by one with each additional category. For example, the IDs for the next landed cost category will be landedcostamount3 and landedcostsource3, and so on.

Using the Payee Address Sublist

As of 2019.2, payee address data is stored in a subrecord instead of a text field. On the Check form in the UI, the address information is now displayed in a sublist instead of a single field. SuiteScript includes specialized APIs that you must use to script with subrecord data. You should use these subrecord APIs to script with payee address data.

For details about scripting subrecords with SuiteScript 2.0, see:

- SuiteScript 2.0 Scripting Subrecords
- Record Object Members
- CurrentRecord Object Members
- Understanding the Address Subrecord

**Warning:** This change may impact existing scripts that reference check payee address data. You should review these scripts to determine whether updates are needed.

Using Profit, Cost, and Amount Fields

The following table provides usage notes for fields on this record related to estimated gross profit and profit percent, estimated extended cost, and amount.
### Field UI Label | Note
---|---
Est. Gross Profit | When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.
| When this field appears on the sublist line level, this field is not scriptable.
Est. Gross Profit Percent | When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.
| When this field appears on the sublist line level, this field is not scriptable.
Est. Extended Cost | When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.
Amount | This field is not available via search or lookup for any transactions.

### Credit Card Charge

**Note:** This topic applies to all versions of SuiteScript.

The credit card charge record is available only when a credit card account exists in the system.

For help working with this record in the UI, see the help topic [Creating Accounts](#).

The internal ID for this record is **creditcardcharge**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

### Supported Script Types

The credit card charge record is scriptable in server SuiteScript only.

### Supported Functions

The credit card charge record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

### Usage Notes

The script user must have the Credit Card permission with Full permission level.
Code Samples

The following sample shows how to create a $30 credit card charge.

```javascript
var charge = record.create({
    type: record.Type.CREDIT_CARD_CHARGE,
    isDynamic: true
});

charge.setValue({
    fieldId: 'entity',
    value: '4'
});
charge.setValue({
    fieldId: 'account',
    value: '128' // Account must be created before (Type = Credit Card)
});
charge.setValue({
    fieldId: 'exchangerate',
    value: '1.00'
});
charge.setValue({
    fieldId: 'postingperiod',
    value: '45'
});
charge.setValue({
    fieldId: 'memo',
    value: 'MEMO'
});
charge.setValue({
    fieldId: 'class',
    value: '1'
});
charge.setValue({
    fieldId: 'department',
    value: '1'
});
charge.setValue({
    fieldId: 'location',
    value: '1'
});

charge.setSublistValue({
    sublistId: 'expense',
    fieldId: 'account',
    line: 1,
    value: '58'
});
charge.setSublistValue({
    sublistId: 'expense',
    fieldId: 'amount',
    line: 1,
    value: '30.00'
});
charge.setSublistValue({
    sublistId: 'expense',
    fieldId: 'amount',
    line: 1,
    value: '30.00'
});
```
Credit Card Refund

**Note:** This topic applies to all versions of SuiteScript.

The credit card refund record is available only when a credit card account exists in the system. For information about creating a credit card account, see the help topic Creating Accounts.

For help working with this record in the UI, see the help topics Issuing Credit Card Refunds and Credit Card Refund Import.

The internal ID for this record is **creditcardrefund**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types

The credit card refund record is scriptable in server SuiteScript only.

Supported Functions

The credit card refund record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes

The script user must have the Credit Card Refund permission with Full permission level.

Code Samples

The following sample shows how to create a $30 credit card refund.

```javascript
var ccRefund = record.create({
  type: record.Type.CREDIT_CARD_REFUND,
  isDynamic: true
});
ccRefund.setValue({
  fieldId: 'entity',
  value: '4'
});
ccRefund.setValue({
  fieldId: 'account',
  value: '128'  // Account must be created before (Type = Credit Card)
});
ccRefund.setValue({
  fieldId: 'exchangerate',
  value: '1.00'
});
ccRefund.setValue({
  fieldId: 'postingperiod',
  value: '45'
});
ccRefund.setValue({
  fieldId: 'memo',
  value: 'MEMO'
});
ccRefund.setValue({
  fieldId: 'class',
  value: '1'
});
ccRefund.setValue({
  fieldId: 'department',
  value: '1'
});
ccRefund.setValue({
```

fieldId: 'location',
value: '1'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'account',
  line: 1,
  value: '58'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'amount',
  line: 1,
  value: '30.00'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'memo',
  line: 1,
  value: 'MEMO2'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'class_',
  line: 1,
  value: '1'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'department',
  line: 1,
  value: '1'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'location',
  line: 1,
  value: '1'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'customer',
  line: 1,
  value: '3'
});
ccRefund.setSublistValue({
  sublistId: 'expense',
  fieldId: 'isbillable',
  line: 1,
  value: 'F'
});
var recordId = ccRefund.save();
Credit Memo

Note: This topic applies to all versions of SuiteScript.

A credit memo transaction decreases the amount a customer owes you. This type of transaction can be used to reverse a charge billed to a customer. If a customer receives a credit memo after having paid an invoice, this memo can be applied to any of the customer’s open or future invoices.

For help working with this record in the UI, see the help topic Customer Credit Memos.

The internal ID for this record is creditmemo.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The credit memo record is scriptable in both client and server SuiteScript.

Supported Functions

The credit card refund record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

NetSuite enables a user to email a copy of a transaction to a customer or another recipient. There are several ways to email a transaction. One way is to open the transaction for viewing and select Actions > Email. Additional approaches for emailing transactions are described in Emailing Transactions.

If you deploy a beforeLoad user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction record. However, if the companywide preference Use Popup for Main Transaction Email Button is selected, a beforeLoad script will not execute when a user selects Activity > Email. In this case, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this form are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the Message record type.

Note that the Use Popup for Main Transaction Email Button preference does not affect other methods of emailing a transaction. For example, a user may email a transaction by using its Communications subtab or by using the Save and Email button. In these cases, a beforeLoad user event script deployed on the transaction still executes, regardless of how the preference is configured. For more information on the Use Popup for Main Transaction Email Button preference, see the help topic Setting Email Preferences.

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.
Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit when this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent when this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost when this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

Customer Deposit

**Note:** This topic applies to all versions of SuiteScript.

A customer deposit transaction records the funds received when a customer makes an advance payment for an order. This payment is recorded in the general ledger as a liability until the goods or services are actually delivered, and does not affect the customer's accounts receivable balance. After the order is filled, the deposit is applied against the invoice.

For help working with this record in the UI, see Customer Deposits.

The internal ID for this record is `customerdeposit`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The customer deposit record is scriptable in both client and server SuiteScript.

**Supported Functions**

The customer deposit record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

The following table provides usage notes for specific fields on this record.
### Customer Payment

**Note:** This topic applies to all versions of SuiteScript.

When a customer makes a payment, record the payment and apply it to the appropriate invoice or cash sale. Applying a payment decreases the amount due and tracks income.

For help working with this record in the UI, see the help topic Customer Payments.

The internal ID for this record is `customerpayment`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

**Important:** When the apply sublist appears on the customer payment record, the Apply subtab is visible, however, all apply sublist data appears on a subtab called Invoice. Also, note that the UI label for the Amt. Due (due) field can also appear as Amount Remaining.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The customer payment record is scriptable in server SuiteScript only.

### Supported Functions

The customer payment record is partially scriptable — it can be updated, deleted, and searched using SuiteScript. It cannot be created or copied.

### Usage Notes

The following table provides usage notes for specific fields on this record.
### Body Fields

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccnumber</td>
<td>Credit Card #</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>isrecurringpayment</td>
<td>Recurring Payment</td>
<td>A value for this field is stored only if the value for paymentmethod is a credit card.</td>
</tr>
</tbody>
</table>

The Deposits sublist is a list sublist. In the NetSuite Help Center, see the help topic List Sublists for information on this sublist type.

## Customer Refund

**Note:** This topic applies to all versions of SuiteScript.

The internal ID for this record is `customerrefund`.

For help working with this record in the UI, see the help topic Customer Refunds.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

## Supported Script Types

The customer refund record is scriptable in server SuiteScript only.

## Supported Functions

The customer deposit record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

## Usage Notes

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccnumber</td>
<td>Credit Card #</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>Field Internal ID</td>
<td>Field UI Label</td>
<td>Note</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

The Deposits sublist is a list sublist. In the NetSuite Help Center, see the help topic List Sublists for information on this sublist type.

**Using the Payee Address Sublist**

As of 2019.2, payee address data is stored in a subrecord instead of a text field. On the Customer Refund form in the UI, address information is now displayed in a sublist instead of a single field. SuiteScript includes specialized APIs that you must use to script with subrecord data. You should use these subrecord APIs to script with payee address data.

For details about scripting subrecords with SuiteScript 2.0, see:

- SuiteScript 2.0 Scripting Subrecords
- Record Object Members
- CurrentRecord Object Members
- Understanding the Address Subrecord

**Warning:** This change may impact existing scripts that reference customer refund payee address data. You should review these scripts to determine whether updates are needed.

**Deposit**

**Note:** This topic applies to all versions of SuiteScript.

You use the deposit record to adjust the balance of an account.

In the UI, you access this record at Transactions > Bank > Make Deposits.

For help working with this record in the UI, see the help topic Deposits.

The internal ID of this record is deposit.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types

The deposit record is scriptable in both client and server SuiteScript. All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Supported Functions

The deposit record cannot be copied, but otherwise it is fully scriptable. It can be created, updated, deleted, and searched in SuiteScript.

Usage Notes

Be aware of the following:

- To successfully add a new deposit record, you must include at least one line in one of the following three sublists: Payments, Other Deposits, or Cash Back.
- If you use the Multi-Book Accounting feature, be aware that the Accounting Book Detail sublist is read-only and not scriptable.

Code Samples

The following sample shows how to create, load, and delete deposit records.

```javascript
function createBankDeposit(cashSaleId){
    var deposit = record.create({
        type: record.Type.DEPOSIT
    });
    deposit.setValue({
        fieldId: 'account',
        value: '1'
    });
    deposit.setValue({
        fieldId: 'subsidiary',
        value: '1'
    });
    deposit.setSublistValue({
        sublistId: 'payment',
        fieldId: 'id',
        line: 1,
        value: cashSaleId
    });
    deposit.setSublistValue({
        sublistId: 'payment',
        fieldId: 'deposit',
        line: 1,
        value: 'T'
    })
}
```
Disabling or Enabling User Filters

When you create or load a deposit record, you can control whether the script respects the filter preferences set by the user on the record's Deposits > Payments subtab. You control this behavior by using the `disablepaymentfilters` initialization flag.

Set the parameter to `true` to ignore the preferences and disable filters, so that all the payments are loaded.

Set the parameter to `false` to indicate that the system should respect the user's preferences and enable the filters. In this case, filters are applied and fewer payments are loaded. If you have a large number of payments, you can set `disablepaymentfilters` to `false` to improve performance. If you do not set a value for the `disablepaymentfilters` parameter, it defaults to false.

The exact way you work with `disablepaymentfilters` varies slightly depending on which version of SuiteScript you are using, as described in the following sections.

Disabling Filters in SuiteScript 2.0

In SuiteScript 2.0, you can interact with `disablepaymentfilters` when you use the `defaultValues` parameter, which is available for the `record.create(options)` and `record.load(options)`.

For example, the following SuiteScript 2.0 snippet would ignore the user's preferences:
```javascript
var newRecord = record.create({
  type: record.Type.DEPOSIT,
  defaultValues: {
    account: '2',
    disablepaymentfilters: true
  }
});
```

### Specifying Payment IDs

When you create a deposit record, you can choose to specify payment IDs. You control this behavior by using the `deposits` initialization parameter.

#### Specifying Payment IDs in SuiteScript 2.0

To specify payment IDs in SuiteScript 2.0, use the `deposits` initialization parameter, which is available for `record.create(options)`. The `deposits` parameter is not available for `record.load(options)`. For example, the following SuiteScript 2.0 snippet would select the payment IDs of 818 and 819:

```javascript
var myDeposit = record.create({
  type: record.Type.DEPOSIT,
  defaultValues: {
    'deposits': '818,819'
  }
});
```

### Deposit Application

**Note:** This topic applies to all versions of SuiteScript.

A deposit application transaction applies a customer deposit against an invoice after the order is complete.

For help working with this record in the UI, see the help topic Applying a Customer Deposit.

The internal ID for this record is `depositapplication`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The deposit application record is scriptable in both client and server SuiteScript.
Supported Functions

The deposit application record cannot be created, but otherwise it is fully scriptable. It can be copied, updated, deleted, and searched in SuiteScript.

Usage Notes

You do not use `record.create(options)` to create a deposit application record. Deposit applications are always created when a customer deposit is applied to an invoice. The application can only be created by applying an open customer Deposit from the deposit sublist of the customer payment. On submit, the backend creates a deposit application in the amount applied.

You can use the `doc` field on the apply sublist of the customer payment to get the internal ID of the deposit or invoice.

Estimate

Note: This topic applies to all versions of SuiteScript.

The internal ID for this record is `estimate`.

For help working with this record in the UI, see the help topics Estimates and Quotes.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The estimate / quote record is scriptable in both client and server SuiteScript.

Supported Functions

The estimate / quote record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes

See the following sections for more details on working with this record:

- Fields
Email Operations

Fields

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

Also note that this record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

The Shipping sublist's Shipping Tax Code field, internal ID shippingtaxcode, appears only if per-line taxes have been set on the Item sublist.

Email Operations

NetSuite enables a user to email a copy of a transaction to a customer or another recipient.

If you deploy a beforeLoad user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction. However, if the companywide preference **Use Popup for Main Transaction Email Button** is enabled, a beforeLoad script will not execute in certain cases, depending on how the email is sent. That is, the script will not execute if the user views the transaction and selects Actions > Email, or edits the transaction and sends the email by using the Communications tab. In these cases, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this window are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the Message record type.

An additional way of emailing a transaction is by editing the transaction and using the Save & Email button. Note that, even if the Use Popup for Main Transaction Email Button preference is enabled, the popup window is not available for emails sent this way. For that reason, a beforeLoad user event script deployed on the transaction will always be triggered by use of the Save & Email button.

For more information on the Use Popup for Main Transaction Email Button preference, see the help topic **Setting Email Preferences**. For details about emailing transactions, see the help topic **Emailing Transactions**.
Expense Report

Note: This topic applies to all versions of SuiteScript.

An expense report transaction records an employee’s expenses for approval and conversion into a bill. The expense total remains in an unapproved expense account and has no accounting impact until the expense is approved by someone with accounting authority. After an expense report is approved, a bill is created and the expense amount is reflected on the books.

This transaction is available when the Estimates feature is enabled at Setup > Company > Enable Features, on the Employees subtab.

For help working with this record in the UI, see the help topic Expense Reporting.

The internal ID for this record is expensereport.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The expense report record is scriptable in both client and server SuiteScript.

Supported Functions

The expense report record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Fulfillment Request

Note: This topic applies to all versions of SuiteScript.

The fulfillment request record represents a request or demand to fulfill a sales order at a fulfillment location. You create a fulfillment request record from a sales order record.

To use this record, you must have the Fulfillment Request feature enabled.

For help working with this record in the UI, see the help topic Fulfillment Requests.

The internal ID for this record is fulfillmentrequest.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
N/record Module

See the SuiteScript Records Browser for all internal IDs associated with this record.

Supported Script Types

The fulfillment request record is scriptable in both client and server SuiteScript.

Supported Functions

This record supports the following functions: read, edit, transform, delete, and search.

Note: The create and copy functions are not supported.

Usage Notes

To create a fulfillment request record, you must have a sales order ID. The Fulfillment Choice and Location must also be set on each line in the sales order. Depending on the selected fulfillment choice (Ship or Store Pickup), NetSuite sets the fulfillment request type to Ship or Store Pickup respectively. If there are multiple lines on the sales order with different fulfillment choices or locations, you must create separate fulfillment requests for each location/fulfillment choice combination. If all lines on the sales order have the same fulfillment choice and location, you create a single fulfillment request. Trying to create a fulfillment request with different fulfillment choices or different locations results in an exception.

For example, if a sales order has two lines, with the fulfillment choice on the first line set to Ship, and the fulfillment choice on the second line set to Store Pickup, you need to create two fulfillment request records.

Note: When you automate the creation of fulfillment requests in your account, NetSuite creates the required number of fulfillment requests automatically.

Fulfillment requests have one of several statuses. Only some statuses can be set with SuiteScript. Other statuses are set by NetSuite and are based on the fulfillment state of the line items in the fulfillment request. The following table lists the possible statuses of the fulfillment request record. See the help topic Fulfillment Request Statuses for more information about fulfillment request statuses.

<table>
<thead>
<tr>
<th>Status Code</th>
<th>Status Name</th>
<th>Settable</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>New</td>
<td>Yes</td>
</tr>
<tr>
<td>B</td>
<td>In Progress</td>
<td>Yes</td>
</tr>
<tr>
<td>C</td>
<td>Cancelled</td>
<td>Yes</td>
</tr>
<tr>
<td>D</td>
<td>Picked</td>
<td>No</td>
</tr>
<tr>
<td>I</td>
<td>Packed</td>
<td>No</td>
</tr>
<tr>
<td>E</td>
<td>Partially Fulfilled</td>
<td>No</td>
</tr>
<tr>
<td>G</td>
<td>Fulfilled with Exceptions</td>
<td>No</td>
</tr>
<tr>
<td>F</td>
<td>Fulfilled</td>
<td>No</td>
</tr>
<tr>
<td>R</td>
<td>Rejected</td>
<td>No</td>
</tr>
</tbody>
</table>
You can add fulfillment request exceptions at the line item level to a fulfillment request. To create a fulfillment request exception, you need to supply a quantity, an exception type, and an exception reason. The exception reasons are different for each exception type. The following table lists the exception types and reasons.

<table>
<thead>
<tr>
<th>Exception Type ID</th>
<th>Exception Type</th>
<th>Exception Reason ID</th>
<th>Exception Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Picking</td>
<td>-1</td>
<td>Inventory mismatch</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-2</td>
<td>Physical damage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-3</td>
<td>Other</td>
</tr>
<tr>
<td>2</td>
<td>Packing</td>
<td>-4</td>
<td>Packing material shortage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-5</td>
<td>Other</td>
</tr>
<tr>
<td>3</td>
<td>Picking up</td>
<td>-6</td>
<td>Too heavy</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-7</td>
<td>Too big</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-8</td>
<td>Wrong color</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-9</td>
<td>Other</td>
</tr>
<tr>
<td>4</td>
<td>Shipping</td>
<td>-10</td>
<td>Shipper can't pick up</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-11</td>
<td>Shipper did not come</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-12</td>
<td>Shipping address issues</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-13</td>
<td>Other</td>
</tr>
</tbody>
</table>

**Code Samples**

**Example 1**: Create a fulfillment request from a sales order. This sample shows how to create a fulfillment request from a sales order. The sales order ID is 321654. Use `record.transform(options)` to transform the sales order record to a fulfillment request.

```javascript
var fulfillmentRequestRecord = record.transform({
    fromType: record.Type.SALES_ORDER,
    fromId: 321654,
    toType: record.Type.FULFILLMENT_REQUEST,
    isDynamic: true
});
var fulfillmentRequestRecordId = fulfillmentRequestRecord.save();
```

**Example 2**: Add a fulfillment exception on a fulfillment request. This sample shows how to add one fulfillment exception reason to an existing fulfillment request. The fulfillment request ID is 221100. The fulfillment exception is for the second line item in the fulfillment request.

```javascript
var fulfillmentRequestRecord = record.load({
    type: 'fulfillmentrequest',
    id: 221100,
    isDynamic: true
});
fulfillmentRequestRecord.selectNewLine({
    sublistId: 'fulfillmentexception'
});```
Example 3: Create fulfillment requests on a sales order with different fulfillment choices. If a sales order has different fulfillment choices or locations at the line level, you first need to get the values of the fulfillment choice and location fields on each line. As you loop through lines in the fulfillment request, you check whether the values on the line are the same as the values on the first line; if they are different, you discard the line.

```javascript
var salesOrderID = 321654;

var fulfillmentRequestRecord = record.transform({
    fromType: record.Type.SALES_ORDER,
    fromId: salesOrderID,
    toType: record.Type.FULLFILLMENT_REQUEST,
    isDynamic: true
});
var fulfillmentRequestLinesCount = fulfillmentRequestRecord.getLineCount({
    sublistId: 'item'
});

while (fulfillmentRequestLinesCount > 0) {
    var initialLocation = null;
    var initialItemFulfillmentChoice = null;
```
for (line = 1; line <= fulfillmentRequestLinesCount; line++) {
    var currentLocation = fulfillmentRequestRecord.getSublistValue({
        sublistId: 'item',
        fieldId: 'location',
        line: line
    });
    var currentLocation = fulfillmentRequestRecord.getSublistValue({
        sublistId: 'item',
        fieldId: 'itemfulfillmentchoice',
        line: line
    });

    if (initialLocation == null) {
        initialLocation = currentLocation;
        initialItemFulfillmentChoice = currentItemFulfillmentChoice;
    }

    if ((initialLocation == currentLocation) && (initialItemFulfillmentChoice == currentItemFulfillmentChoice)) {
        fulfillmentRequestRecord.setSublistValue({
            sublistId: 'item',
            fieldId: 'itemreceive',
            line: line,
            value: true
        });
    } else {
        // Discard the line because its location or fulfillment choice is different
        fulfillmentRequestRecord.setSublistValue({
            sublistId: 'item',
            fieldId: 'itemreceive',
            line: line,
            value: false
        });
    }
}

fulfillmentRequestRecordId = fulfillmentRequestRecord.save();

var fulfillmentRequestRecord = record.transform({
    fromType: record.Type.SALES_ORDER,
    fromId: salesOrderID,
    toType: record.Type.FULLFILLMENT_REQUEST,
    isDynamic: true
});

var fulfillmentRequestLinesCount = fulfillmentRequestRecord.getLineCount({
    sublistId: 'item'
});
GL Audit Numbering Sequence

Note: This topic applies to all versions of SuiteScript.

The GL Audit Numbering feature enables you to set up and run two types of GL audit numbering sequences: permanent and repeatable. When you run a permanent GL audit numbering sequence, the number assigned to a GL impacting transaction cannot be changed. You can, however, modify the details of the original GL impacting transaction. When you run a repeatable GL audit numbering sequence, you can re-run the numbering sequence on GL impacting transactions as often as required to address gaps in numbering that might occur due to adjustments made to your GL.

Note: Re-running a repeatable numbering sequence renumbers transactions and may assign a number different from the previously assigned number.

In the UI, you can access this record at Transactions > Management > GL Audit Numbering Sequences > New. This page includes fields to set up a GL audit numbering system, including the sequence name, numbering type, available subsidiaries, and the numbering format.

For help working with this record in the UI, see the help topic Setting Up a GL Audit Numbering Sequence.

To use this record in scripts, you must be using the GL Audit Numbering feature and the Manage Accounting Period permission.

Note: The System Notes sublist and the Next Run field in the Recurring tab for Permanent Numbering are always search-only.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The GL audit numbering sequence record is scriptable in both client and server SuiteScript.

Supported Functions

The GL audit numbering sequence record can be read, created, updated, copied, and searched using SuiteScript.

Code Samples

The following sample shows how to create a new GL audit numbering sequence. The sequencename, period, subsidiaries, numberingtype, accountingbook, initnum, ordertype, and recurringtype fields must all be defined.

```javascript
var newGlSequence = record.create({
    type: record.Type.GL_NUMBERING_SEQUENCE
});
```
newGlSequence.setValue({
    fieldId: 'sequencename',
    value: 'My sequence'
});
newGlSequence.setValue({
    fieldId: 'period',
    value: 1 // The ID of the accounting period to which this sequence belongs
});
newGlSequence.setValue({
    fieldId: 'subsidiaries',
    value: 1 // The ID of the subsidiary to which this sequence belongs
});
newGlSequence.setValue({
    fieldId: 'numberingtype',
    value: 'REPEATABLE' // Can be set to either 'REPEATABLE' or 'PERMANENT'
});
newGlSequence.setValue({
    fieldId: 'accountingbook',
    value: 1 // ID of the accounting book to which this sequence belongs
});
newGlSequence.setValue({
    fieldId: 'initnum',
    value: 1 // Typically 1 - the starting number of GL numbering
});
newGlSequence.setValue({
    fieldId: 'ordertype',
    value: 'CREATEDDATE' // Can be set to either 'CREATEDDATE' or 'TRANDATE'
});
newGlSequence.setValue({
    fieldId: 'recurringtype',
    value: 'NONE' // Can be set to either 'NONE', 'HOURLY', or 'DAILY'
});
var id = newGlSequence.save(); // On submission, the system will check all the rules and constraints on
// mandatory fields and produce an error if something is incorrect.

Intercompany Journal Entry

**Note:** This topic applies to all versions of SuiteScript.

Intercompany journal entries are a specialized type of journal available specifically for OneWorld. An intercompany journal entry records debits and credits to be posted to ledger accounts for transactions between two subsidiaries. These records adjust the value of any set of accounts without the need for transactions such as invoices and bills.

In the UI, you can access this record at Transactions > Financial > Make Intercompany Journal Entries.

If your account has the Multi-Book Accounting feature enabled, you can also work with book specific intercompany journal entry records, which in the UI are available at Transactions > Financial > Make Book Specific Intercompany Journal Entries. Although they have different entry forms, both book specific and regular intercompany journal entries are the same record type. Within SuiteScript, they are differentiated by the accountingbook field. In other words, a record that has a value set for accountingbook is book specific. Otherwise, the record is a regular intercompany journal entry.

For help working with this record in the UI, see the help topics Making Intercompany Journal Entries and Book-Specific Intercompany Journal Entries.
The internal ID for this record is **`intercompanyjournalentry`**. See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The intercompany journal entry record is scriptable in both client and server SuiteScript. All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

The intercompany journal entry record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Warning:** If you update any type of journal entry that has been applied as a payment to an invoice or vendor bill, the relationship between the journal entry and payment is removed and the payment is no longer applied.

**Code Samples**

The following sample shows how to create a book specific intercompany journal entry. The record is book specific because a value has been set for the `accountingbook` field.

```javascript
var journalEntry = record.create({
    type: record.Type.INTER_COMPANY_JOURNAL_ENTRY,
    isDynamic: true
});
journalEntry.setValue({
    fieldId: 'accountingbook',
    value: 2                      // Setting a value for this field makes the record book-specific.
});
journalEntry.setValue({
    fieldId: 'subsidiary',
    value: 1
});
journalEntry.setValue({
    fieldId: 'tosubsidiary',
    value: 3
});
journalEntry.selectNewLine({
    sublistId: 'line'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'linesubsidiary',
```
transactions

value: 1
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: 1
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'credit',
    value: '2.00'
});
journalEntry.commitLine({
    sublistId: 'line'
});
journalEntry.selectNewLine({
    sublistId: 'line'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'linesubsidiary',
    value: 1
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: 2
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'debit',
    value: '2.00'
});
journalEntry.commitLine({
    sublistId: 'line'
});
journalEntry.selectNewLine({
    sublistId: 'line'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'linesubsidiary',
    value: 3
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: 6
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'credit',
    value: '2.00'
});
Intercompany Transfer Order

**Note:** This topic applies to all versions of SuiteScript.

In accounts using NetSuite OneWorld and the Multi-Location Inventory (MLI) feature, you can use the Intercompany Transfer Order transaction to move inventory from a location for one subsidiary to a location for another subsidiary.

In the UI, you can access this record at Transactions > Inventory > Enter Intercompany Transfer Orders.

For help working with this record in the UI, see the help topic Intercompany Inventory Transfers - Non-Arm's Length.

The internal ID for this record is `intercompanytransferorder`.

See the [SuiteScript Records Browser](https://www.netsuite.com) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types

The intercompany transfer order record is scriptable in both client and server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Supported Functions

The intercompany transfer order record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes

Intercompany transfer orders are only available when transfer pricing is used, meaning the Use Item Cost as Transfer Cost, at Setup > Accounting > Preferences > Accounting Preferences (Administrator), on the Order Management subtab, must be disabled.

This record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

Code Samples

The following sample shows how to create an intercompany transfer order.

```javascript
var incoTransfer = record.create({
  type: record.Type.INTER_COMPANY_TRANSFER_ORDER,
  isDynamic: true
});
incoTransfer.setValue({
  fieldId: 'orderstatus',
  value: expectedValues.get(orderstatus) //expectedValues is previously defined
});
incoTransfer.setValue({
  fieldId: 'subsidiary',
  value: expectedValues.get(subsidiary) //expectedValues is previously defined
});
incoTransfer.setValue({
  fieldId: 'tosubsidiary',
  value: expectedValues.get(tosubsidary) //expectedValues is previously defined
});
incoTransfer.setText({
  fieldId: 'location',
  text: sourceLocationName1
});
incoTransfer.setText({
  fieldId: 'transferlocation',
  text: destinationLocationName1
});
incoTransfer.selectNewLine({
  sublistId: 'item'
});
incoTransfer.setCurrentSublistValue({
```

Transactions | 140
```javascript
sublistId: 'item',
fieldId: 'Item.item',
value: itemKey
});
incoTransfer.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'Item.rate',
    value: rate
});
incoTransfer.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'quantity',
    value: quantity
});
incoTransfer.commitLine({
    sublistId: 'item'
});
var key = incoTransfer.save();
```

Inventory Adjustment

**Note:** This topic applies to all versions of SuiteScript.

The inventory adjustment transaction changes the quantity and value of an inventory item without entering a purchase order. For example, this transaction can be used to account for clerical errors, changes in cost, thefts, or miscounts. If you use the LIFO or FIFO costing methods, this transaction preserves the quantity and value of an inventory item at the same time preserving the costing history of the item.

For help working with this record in the UI, see the help topic Adjusting Inventory.

The internal ID for this record is `inventoryadjustment`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The inventory adjustment record is scriptable in both client and server SuiteScript.

**Supported Functions**

The inventory adjustment record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Inventory Cost Revaluation

Note: This topic applies to all versions of SuiteScript.

The inventory cost revaluation record is used to recalculate the value of items configured to use standard costing.

This record is available when the Standard Costing feature is enabled at Setup > Company > Enable Features, on the Items & Inventory subtab. When the feature is enabled, you can access the inventory cost revaluation record in the UI by choosing Transactions > Inventory > Revalue Inventory Cost.

For help working with this record in the UI, see the help topics Manually Enter an Inventory Cost Revaluation and Revaluing Standard Cost Inventory.

The internal ID for this record is inventorycostrevaluation.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The inventory cost revaluation record is scriptable in both client SuiteScript and Server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Supported Functions

The inventory cost revaluation record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Code Samples

The following sample shows how to create and update inventory cost revaluation records.

```javascript
function inventoryCostRevaluation(){
  var recId = null;
  var newId = null;
  var recType = record.Type.INVENTORY_COST_REVALUATION;
  try {
    // Create the record
    var revaluationRec = record.create({
      type: recType,
      isDynamic: true
    });
    revaluationRec.setValue({
      fieldId: 'subsidiary',
      value: '1'
    });
    revaluationRec.setValue({
      fieldId: 'item',
      value: '3'  // or whatever value you need
    });
  } catch (err) {
    // Handle error
  }
}
```
```javascript
value: itemId1 // Some assembly item
});
revaluationRec.setValue({
    fieldId: 'account',
    value: '1'
});
revaluationRec.setValue({
    fieldId: 'location',
    value: '1'
});
revaluationRec.setSublistValue({
    sublistId: 'costcomponent',
    fieldId: 'cost',
    line: 1,
    value: '2'
});
revaluationRec.setSublistValue({
    sublistId: 'costcomponent',
    fieldId: 'componentitem',
    line: 1,
    value: componentItemId1 // Some inventory item
});
revaluationRec.setSublistValue({
    sublistId: 'costcomponent',
    fieldId: 'quantity',
    line: 1,
    value: '3'
});
recId = revaluationRec.save();

var revaluationAddedRec = recordload({
    type: recType,
    id: recId
});

// Update
// Note you cannot change subsidiary and item
revaluationAddedRec.setValue({
    fieldId: 'account',
    value: '2'
});
revaluationAddedRec.setSublistValue({
    sublistId: 'costcomponent',
    fieldId: 'quantity',
    line: 1,
    value: '5'
});
recId2 = revaluationAddedRec.save();

var revaluationUpdatedRec = record.load({
    type: recType,
    id: recId2
});
```
// Copy
var copiedRecord = record.copy({
  type: recType,
  id: recId
});
var newId = copiedRecord.save();
copiedRecord = record.load({
  type: recType,
  id: newId
});

var mySearch = search.create({
  type: recType,
  columns: [{
    name: 'memo',
  }, {
    name: 'subsidiary'
  }],
  filters: [{
    name: 'memo',
    operator: 'contains',
    value: 'em'
  }]
});

log.debug({
  details: arguments.callee.name + ' passed.'
});
}
catch (e) {
  log.error({
    details: arguments.callee.name + ' failed.'
  });
}
finally{
  if (recId != null){
    record.delete({
      type: recType,
      id: recId
    });
  }
  if (newId != null){
    record.delete({
      type: recType,
      id: newId
    });
  }
}
Inventory Count

Note: This topic applies to all versions of SuiteScript.

The inventory count record enables you to maintain better inventory accuracy and tighter control of assets.

The inventory count record is available only when the Inventory Count feature is enabled at Setup > Enable Features, on the Items & Inventory subtab.

In the UI, you access the inventory count record at Transactions > Inventory > Enter Inventory Count.

For help working with this record in the UI, see the help topic Inventory Count.

The internal ID for this record is `inventorycount`.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The inventory count record is scriptable in both client and server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Supported Functions

The inventory count record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

You cannot use SuiteScript to update the Status field on inventory count records. You must set this field by clicking the Start Count button manually on the inventory count record in the UI.

Code Samples

The following samples show how to create an inventory count record with an associated inventory detail record for a lot-numbered or serialized item.

It includes the following:
- The code in Step 1 creates an inventory count record. This code is supported for client or server scripts.
- The code in Step 2 adds an Add Inventory Detail button to the record page before it is loaded. This code is supported for Suitelets, beforeLoad user event scripts, and workflow action scripts. A click of the Add Inventory button calls a client script.
- The code in Step 3 uses an https request to call a Suitelet or RESTlet and refreshes the page. This code is supported for client scripts.
The code in Step 4 adds an inventory detail subrecord to the inventory count. This code is supported for Suitelets and RESTlets.

**Step 1:** The following sample creates the inventory count record in dynamic mode.

The following sample shows how to create and update inventory cost revaluation records.

```javascript
/**
 * @NApiVersion 2.x
 */
define(["N/record"], function(record) {
    // Script parameters:
    var subsidiaryId = 1;
    var locationId = 6;
    var itemId = 599;
    var binNumber = 501;

    var inventoryCount = record.create({
        type: record.Type.INVENTORY_COUNT,
        isDynamic: true
    });
    inventoryCount.setValue({
        fieldId: 'subsidiary',
        value: subsidiaryId
    });
    inventoryCount.setValue({
        fieldId: 'location',
        value: locationId
    });
    inventoryCount.selectNewLine({
        sublistId: 'item'
    });
    inventoryCount.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'item',
        value: itemId
    });
    inventoryCount.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'binnumber',
        value: binNumber
    });
    inventoryCount.commitLine({
        sublistId: 'item'
    });
    var recordId = inventoryCount.save();
    log.debug(
        {
            title: 'Inventory Count Created: ' + inventoryCount
        }
    );
});
```

**Step 2:** The following sample adds an Add Inventory Detail button to the record page before it is loaded.

```javascript
/**
 * @NApiVersion 2.x
 */
define(["N/record"], function(record) {
    // Script parameters:
    var subsidiaryId = 1;
    var locationId = 6;
    var itemId = 599;
    var binNumber = 501;

    var inventoryCount = record.create({
        type: record.Type.INVENTORY_COUNT,
        isDynamic: true
    });
    inventoryCount.setValue({
        fieldId: 'subsidiary',
        value: subsidiaryId
    });
    inventoryCount.setValue({
        fieldId: 'location',
        value: locationId
    });
    inventoryCount.selectNewLine({
        sublistId: 'item'
    });
    inventoryCount.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'item',
        value: itemId
    });
    inventoryCount.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'binnumber',
        value: binNumber
    });
    inventoryCount.commitLine({
        sublistId: 'item'
    });
    var recordId = inventoryCount.save();
    log.debug(
        {
            title: 'Inventory Count Created: ' + inventoryCount
        }
    );
});
```
Step 3: The following sample uses an https request to call a Suitelet or RESTlet that will add the inventory detail, and refreshes the page.
Step 4: The following sample adds the inventory detail subrecord to the inventory count record.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Restlet
 *
 * Expects parameters: (configure these in the script deployment)
 *   custscriptcountquantity
 *   custscriptinventorynumber
 */
define(["N/error", "N/record", "N/runtime"], function(error, record, runtime) {
    function put(requestBody) {
        // Script parameters
        var script = runtime.getCurrentScript();
        var countQuantity = script.getParameter({
            name: 'custscriptcountquantity'
        });
        var inventoryNumber = script.getParameter({
            name: 'custscriptinventorynumber'
        });
```
log.debug({
    details: "Count Quantity: " + countQuantity
});
log.debug({
    details: "Inventory Number: " + inventoryNumber
});

var inventoryCount = record.load({
    type: 'inventorycount',
    id: requestBody.inventoryCountId
});

var itemLineCount = inventoryCount.getLineCount({
    sublistId: 'item'
});
for (i = 0; i < itemLineCount; i++) {
    inventoryCount.setSublistValue({
        sublistId: 'item',
        fieldId: 'countquantity',
        line: i,
        value: countQuantity
    });
    inventoryCount.getSublistSubrecord({
        sublistId: 'item',
        fieldId: 'countdetail',
        line: i
    });
    inventoryCount.insertLine({
        sublistId: 'inventorydetail',
        line: 0
    });
    inventoryCount.setSublistValue({
        sublistId: 'inventorydetail',
        fieldId: 'inventorynumber',
        value: inventoryNumber,
        line: 0
    });
    inventoryCount.setSublistValue({
        sublistId: 'inventorydetail',
        fieldId: 'quantity',
        value: countQuantity,
        line: 0
    });
    var recordId = inventoryCount.save();
}

return {
    put: put
};
Inventory Detail

**Note:** This topic applies to all versions of SuiteScript.

The inventory detail subrecord is available when the Advanced Bin / Numbered Inventory Management feature is enabled. This type of record stores values relating to bin numbers and/or serial/lot numbers for items, including line items on transactions. This data includes the quantity on hand and quantity available per bin number, or per serial/lot number, or when both are in use, per bin number/serial lot number combination.

In the UI, the inventory detail subrecord displays as a popup when you click the Inventory Detail body field or sublist field.

For help working with this record in the UI, see the help topic Advanced Bin / Numbered Inventory Management.

The internal ID for this subrecord is `inventorydetail`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The inventory detail record is scriptable in client SuiteScript. Server scripts must access through the parent record.

**Supported Functions**

The inventory detail record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

Inventory Detail is scriptable from both the body field and the line item.

Inventory Detail is considered a subrecord in SuiteScript. For details on working with this subrecord type, see the help topic Scripting the Inventory Detail Subrecord. For general details on working with subrecords, see the help topic Working with Subrecords in SuiteScript.

For information on adding an inventory detail subrecord to an inventory count record, see Creating an Inventory Count Record Using SuiteScript 2.0.

**Inventory Status Change**

**Note:** This topic applies to all versions of SuiteScript.

The inventory status change record is available in NetSuite and NetSuite OneWorld accounts when the Inventory Status feature is enabled. An administrator can enable this feature at Setup > Company > EnableFeatures, on the Items & Inventory subtab.
For help working with this record in the UI, see the help topic Inventory Status.
The internal ID for the inventory status change record is inventorystatuschange. The internal ID for its Items sublist is inventory.
See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**
The inventory status change record is scriptable in both client and server SuiteScript.

**Supported Functions**
The inventory status change record can be created, read, edited, copied, deleted, searched, and transformed using SuiteScript.

**Usage Notes**
In the UI, you cannot select the same value in the Previous State and Revised State fields. Using SuiteScript, there is no validation if you set the same value for previousstatus and revisedstatus.

**Code Samples**
The following sample shows how to create an inventory status change record to move 10 good widgets at your US location to the inspection status.

```javascript
var invStatus = record.create({
    type: record.Type.INVENTORY_STATUS_CHANGE,
    isDynamic: true
});
invStatus.setText({
    fieldId: 'location',
    text: 'US Only Location'
});
invStatus.setText({
    fieldId: 'previousstatus',
    text: 'Good'
});
invStatus.setText({
    fieldId: 'revisedstatus',
    text: 'Inspection'
});
invStatus.selectNewLine({
    sublistId: 'inventory'
});
invStatus.setCurrentSublistText({
    sublistId: 'inventory'
});
```
Inventory Transfer

**Note:** This topic applies to all versions of SuiteScript.

The Inventory Transfer transaction posts details about per-location item inventory level changes when items are transferred between two locations. This basic inventory transfer decreases items in the source location and increases them in the receiving location, all in one step.

This transaction is available when the Locations feature and the Multi-Location Inventory (MLI) feature are enabled.

For help working with this record in the UI, see the help topic Transferring Inventory.

The internal ID for this record is `inventorytransfer`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The inventory transfer record is scriptable in server SuiteScript only.

**Supported Functions**

The inventory detail record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

**Usage Notes**

This record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.
Invoice

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Invoices.

The internal ID for this record is invoice.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The invoice record is scriptable in both client and server SuiteScript.

Supported Functions

The invoice record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes

See the following sections for more details on working with this record:

- Fields
- Email Operations

Fields

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrosprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrosprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>
This record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic record.transform(options) for examples on how to transform records.

The following invoice record sublists are list sublists:

- Billable Expenses
- Billable Items
- Billable Time

In the NetSuite Help Center, see the help topic List Sublists for information on this sublist type.

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

The Shipping sublist’s Shipping Tax Code field, internal ID shippingtaxcode, appears only if per-line taxes have been set on the Item sublist.

**Email Operations**

NetSuite enables a user to email a copy of a transaction to a customer or another recipient. If you deploy a beforeLoad user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction. However, if the companywide preference Use Popup for Main Transaction Email Button is enabled, a beforeLoad script will not execute in certain cases, depending on how the email is sent. That is, the script will not execute if the user views the transaction and selects Actions > Email, or edits the transaction and sends the email by using the Communications tab. In these cases, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this window are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the Message record type.

An additional way of emailing a transaction is by editing the transaction and using the Save & Email button. Note that, even if the Use Popup for Main Transaction Email Button preference is enabled, the popup window is not available for emails sent this way. For that reason, a beforeLoad user event script deployed on the transaction will always be triggered by use of the Save & Email button.

For more information on the Use Popup for Main Transaction Email Button preference, see the help topic Setting Email Preferences. For details about emailing transactions, see the help topic Emailing Transactions.

**Item Demand Plan**

*Note:* This topic applies to all versions of SuiteScript.

The item demand plan record is available for inventory items when the Demand Planning feature is enabled, and for assembly/BOM items when both the Demand Planning and Work Orders features are enabled. Item demand plans can be added for items where the supplyReplenishMethod field is set to Time Phased.

An item demand plan transaction stores the quantity expected to be needed, during specified time periods, for an item. NetSuite supports three types of demand plans: monthly, weekly, and daily.

For help working with this record in the UI, see the help topic Demand Planning.

The internal ID for this record is itemdemandplan.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Transactions

For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The item demand plan record is scriptable in both client and server SuiteScript.

Supported Functions

The invoice record is partially scriptable — it can be created, updated, deleted, and searched using SuiteScript. It cannot be copied or transformed.

Usage Notes

A demand plan records the expected future demand for an item based on previous or projected demand. When the Demand Planning feature is enabled, demand plans can be created for inventory items. When the Work Orders feature is also enabled, demand plans also can be created for assembly/BOM items. Demand plans can only be created for items that have a value of "Time Phased" for the supplyreplenishment method field.

Each demand plan record includes:
- A set of body fields used to uniquely identify the demand plan, define the time period it covers, and indicate the time period it uses (monthly, weekly, or daily).
  - Body fields must be defined before matrix field values can be edited.
- A matrix of projected quantities per time period, similar to the matrix used for item pricing.
  - In a monthly demand plan, this matrix includes a row for each month in the time period, and one column with the projected quantity demand for each month.
  - In a weekly demand plan, this matrix includes a row for each week in the time period, and one column with the projected quantity demand for each week.
  - In a daily demand plan, this matrix includes a row for each week in the time period and seven columns with the projected quantity demand for each day of each week.
- Review the following table for details about Item Demand Plan body and matrix sublist fields. For more details and code samples, see Demand Plan Detail Sublist.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>subsidiary</td>
<td>Subsidiary</td>
<td>Required in OneWorld accounts.</td>
</tr>
<tr>
<td>location</td>
<td>Location</td>
<td>Required when the Multi-Location Inventory feature enabled.</td>
</tr>
<tr>
<td>item</td>
<td>Item</td>
<td>Required. Can only use items with supplyreplenishment method set to Time Phased.</td>
</tr>
<tr>
<td>units</td>
<td>Unit of Measure</td>
<td>Optional.</td>
</tr>
</tbody>
</table>
### Field Internal ID | Field UI Label | Note
--- | --- | ---
memo | Memo | Available when the Multiple Units of Measure feature is enabled.
startdate | Start Date | Optional. Defaults to the first day of the current year, for example for 2011, defaults to 1/1/2011.
enddate | End Date | Optional. Defaults to the last day of the current year, for example for 2011, defaults to 12/31/2011.
demandplancalendartype | View | Required. Valid values are MONTHLY, WEEKLY, or DAILY. (Must use all capital letters.)

#### Matrix Fields
- **quantity**
  - **Quantity**
  - For monthly and weekly demand plans, each row has one quantity column.
  - For daily demand plans, each row has seven quantity columns.
- **startdate**
  - **Start Date**
  - System-calculated, read-only values.
  - For monthly plans, the date of the first day of the month that the row represents.
  - For weekly and daily plans, the date of the first day of the week that the row represents, based on the preference set for First Day of Week at Setup > Company > General Preferences.
- **enddate**
  - **End Date**
  - System-calculated, read-only values.
  - For monthly plans, the date of the last day of the month that the row represents.
  - For weekly and daily plans, the date of the last(seventh) day of the week that the row represents, based on the preference set for First Day of Week at Setup > Company > General Preferences.

**Note:** It is recommended that you work with the Item Demand Plan record in dynamic mode. See the help topic **SuiteScript 2.0 – Standard and Dynamic Modes.**

### Demand Plan Detail Sublist

**Note:** This topic applies to all versions of SuiteScript.

The internal ID for this sublist is **demandplandetail**.

The Demand Plan Detail sublist appears on the **Item Demand Plan** record type.

The Demand Plan Detail sublist is a matrix that is similar to the Pricing sublist. This matrix stores projected quantities demanded by date. Each row in the matrix represents a specific month, week, or day, and each column in the matrix represents an expected quantity demand.

For help working with this record in the UI, see the help topic **Demand Planning**.

Functionally, this sublist shares many of the characteristics of List Sublists. However, scripting with the Demand Plan Detail sublist is not like scripting with most other sublists in NetSuite. You must use **Matrix APIs for the Demand Plan Detail Sublist** to access quantity values on a per-row, per-column basis, similar...
to the way that item pricing values are accessed. These APIs are a subset of the `N/currentRecord` Module and `N/record` Module APIs more commonly used for scripting with other sublists.

The format of the Demand Plan Detail sublist depends on the values set in body fields for the start date of the plan, the end date of the plan, and the time period to be used (monthly, weekly, or daily). Because of this dependence, it is recommended that you work with the Item Demand Plan record and the Demand Plan Detail sublist in dynamic mode. See the help topic SuiteScript 2.0 – Standard and Dynamic Modes

Be aware of the following requirements:

- To script with the Item Demand Plan record and the Demand Plan Detail sublist for inventory items, the Demand Planning feature must be enabled. For assembly/BOM items, the Work Orders feature also must be enabled.
- Demand plans are supported only for item(s) that have the `supplyreplenishmentmethod` field set to `Time Phased`.
- Required body field values must be defined before matrix field values can be edited. In dynamic mode, current values may be retrieved. Start date and end date body fields default to the first day and last day of the current year.

For more details and code samples for each type of demand plan, see the following help topics:

- Monthly Demand Plan
- Weekly Demand Plan
- Daily Demand Plan

## Monthly Demand Plan

A monthly demand plan includes a row for each month within the body field start date and end date, and one quantity column for each month.
The sublist startdate and enddate fields are system-calculated and read-only.

- The startdate is the date of the first day of the month represented by each row.
- The enddate is the date of the last day of the month represented by each row.
- The month for row 1 is the month set in the body field start date, the month for row 2 is the next month, and so on, until the month set in the body field end date is reached.
- The values for the quantity field can be set in SuiteScript. For monthly demand plans, the column parameter for this field is always 1.

**Monthly Demand Plan Code Sample**

The following sample sets quantities for the months of January and February, 2011:

```javascript
var demandplan = record.create(
    type: record.Type.ITEM_DEMAND_PLAN,
    isDynamic: true
));

demandplan.setValue({
    fieldId: 'demandplancalendartype',
    value: 'MONTHLY'
});

demandplan.setValue({
    fieldId: 'subsidiary',
    value: 1
});
demandplan.setValue({
    fieldId: 'location',
    value: 1
});
demandplan.setValue({
    fieldId: 'item',
    value: 165
});
demandplan.setValue({
    fieldId: 'startdate',
    value: '1/1/2015'
});
demandplan.setValue({
    fieldId: 'enddate',
    value: '12/31/2015'
});

demandplan.selectLine({
    sublistId: 'demandplandetail',
    line: '1'
});
demandplan.setCurrentMatrixSublistValue({
    sublistId: 'demandplandetail',
    fieldId: 'quantity',
    column: '1',
    value: 100
});
```
Weekly Demand Plan

A weekly demand plan includes a row for each week contained in the time period set by the body field start date and end date, and one quantity column for each week.

- The sublist startdate and enddate fields are system-calculated and read-only.
  - The startdate is the date of the first day of the week represented by each row.
  - The enddate is the date of the last day of the week represented by each row.
Note: The first day of the week by default is Sunday, but may vary according to the company preference set for First Day of the Week at Setup > Company > General Preferences.

- The week for row 1 is the week of the date set in the body field start date. Note that unless the body field start date happens to be the first day of the week, the startdate for this first row may precede the body field start date.
- The week for the final sublist row is the week of the date set in the body field end date. Note that unless the body field end date happens to be the last day of the week, the enddate for this last row may be after the body field enddate.
- The values for the quantity field can be set in SuiteScript. For weekly demand plans, the column parameter for this field is always 1.

Weekly Demand Plan Code Sample

The following sample sets quantities for the first two weeks of 2011:

```javascript
var demandPlan = record.create({
    type: record.Type.ITEM_DEMAND_PLAN,
    isDynamic: true
});
demandPlan.setValue({
    fieldId: 'demandplancalendartype',
    value: 'WEEKLY'
});
demandPlan.setValue({
    fieldId: 'subsidiary',
    value: 1
});
demandPlan.setValue({
    fieldId: 'location',
    value: 1
});
demandPlan.setValue({
    fieldId: 'item',
    value: 165
});
demandPlan.setValue({
    fieldId: 'startdate',
    value: '1/1/2015'
});
demandPlan.setValue({
    fieldId: 'enddate',
    value: '12/31/2015'
});
demandPlan.selectLine({
    sublistId: 'demandplandetail',
    line: '1'
});
demandPlan.setCurrentMatrixSublistValue({
    sublistId: 'demandplandetail',
    // Code continues here...
});
```
Daily Demand Plan

A daily demand plan includes a row for each week contained in the time period set by the body field start date and end date, and seven quantity columns for each week, one for each day of the week.
The sublist startdate and enddate fields are system-calculated and read-only.

- The startdate is the date of the first day of the week represented by each row.
- The enddate is the date of the last day of the week represented by each row.

**Note:** The first day of the week by default is Sunday, but may vary according to the company preference set for First Day of the Week at Setup > Company > General Preferences.

- The week for row 1 is the week of the date set in the body field start date. Note that unless the body field start date happens to be the first day of the week, the startdate for this first row may precede the body field start date.
- The week for the final sublist row is the week of the date set in the body field end date. Note that unless the body field end date happens to be the last day of the week, the enddate for this last row may be after the body field enddate.

The values for the quantity fields can be set in SuiteScript.

- The column parameter for a quantity field is 1,2,3,4,5,6, or 7, depending upon the day of the week.
- In the screenshot above, the week starts with Sunday, which is the default first day of the week, and in this case, maps to a column parameter of 1. However, 1 does not always map to Sunday; it maps to the first day of the week as set in the company preferences.

**Daily Demand Plan Code Sample**

```javascript
var demandPlan = record.create({
    type: record.Type.ITEM_DEMAND_PLAN,
    isDynamic: true
});
demandPlan.setValue({
    fieldId: 'demandplancalendartype',
    value: 'DAILY'
});
demandPlan.setValue({
    fieldId: 'subsidiary',
    value: 1
});
demandPlan.setValue({
    fieldId: 'location',
    value: 1
});
demandPlan.setValue({
    fieldId: 'item',
    value: 165
});
demandPlan.setValue({
    fieldId: 'startdate',
    value: '1/1/2011'
});
demandPlan.setValue({
    fieldId: 'enddate',
    value: '12/31/2011'
});

demandPlan.selectLine({
```
Matrix APIs for the Demand Plan Detail Sublist

Use the following matrix APIs with the Demand Plan Detail sublist:

- `Record.getCurrentMatrixSublistValue(options)`
- `CurrentRecord.getCurrentMatrixSublistValue(options)`
- `Record.setCurrentMatrixSublistValue(options)`
- `CurrentRecord.setCurrentMatrixSublistValue(options)`

```javascript
sublistId: 'demandplandetail',
line: '1' // week of 12/26/2010 to 1/1/2011
});
demandPlan.setCurrentMatrixSublistValue({
  sublistId: 'demandplandetail',
  fieldId: 'quantity',
  column: '1',
  value: 100 // sunday
});
demandPlan.setCurrentMatrixSublistValue({
  sublistId: 'demandplandetail',
  fieldId: 'quantity',
  column: '2',
  value: 101 // monday
});
demandPlan.setCurrentMatrixSublistValue({
  sublistId: 'demandplandetail',
  fieldId: 'quantity',
  column: '3',
  value: 102 // tuesday
});
demandPlan.selectLine({
  sublistId: 'demandplandetail',
  line: '2' // week of 1/2/2011 to 1/8/2011
});
demandPlan.setCurrentMatrixSublistValue({
  sublistId: 'demandplandetail',
  fieldId: 'quantity',
  column: '1',
  value: 200 // sunday
});
demandPlan.setCurrentMatrixSublistValue({
  sublistId: 'demandplandetail',
  fieldId: 'quantity',
  column: '5',
  value: 200 // thursday
});

var id = record.save({
  enableSourcing: true
});
```
Note: With all APIs listed above, use the `Record.selectLine(options)` or `CurrentRecord.selectLine(options)` APIs first to select an existing line.

- `Record.getMatrixSublistField(options)`
- `CurrentRecord.getMatrixSublistField(options)`
- `Record.getMatrixSublistValue(options)`
- `CurrentRecord.getMatrixSublistValue(options)`
- `Record.findMatrixSublistLineWithValue(options)`
- `CurrentRecord.findMatrixSublistLineWithValue(options)`

For more information about APIs, see the help topics `N/currentRecord Module` and `N/record Module`.

Item Fulfillment

Note: This topic applies to all versions of SuiteScript.

An item fulfillment transaction records the shipment of some or all items on an order to the customer. The processes for item fulfillment transactions depend on whether the Advanced Shipping feature is enabled.

- If Advanced Shipping is not enabled, the fulfillment and invoicing processes are combined. When an item fulfillment is created, a related invoice is created at the same time.
- If Advanced Shipping is enabled, fulfillment and invoicing are two independent processes, and shipments can be recorded separately from billing.

For help working with this record in the UI, see the help topic Order Fulfillment.

The internal ID for this record is `itemfulfillment`. Copy and create are not allowed for this record.

For information about scripting with this record in SuiteScript, see the following help topics:

- `SuiteScript 2.0 Scripting Records and Subrecords`
- `SuiteScript 2.0 Custom Forms`
- `N/record Module`

Supported Script Types

The item fulfillment record is scriptable in both client and server SuiteScript.

Supported Functions

The item fulfillment record is partially scriptable — it can be updated, deleted, and searched using SuiteScript. It cannot be created or copied.

Usage Notes

When working with this record, you can set `pick`, `pack`, or `ship` as event trigger types that will execute your user event script. In the NetSuite Help Center, see the help topic User Event Script Execution Types for more information.
The Item Fulfillment/Item Receipt sublist is a **list** sublist. In the NetSuite Help Center, see the help topic [List Sublists](https://www.netsuite.com/standards/suitelet/suitelet-development/list-sublists) for information on this sublist type.

The Shipping sublist is included on the item fulfillment record. This sublist is not currently showing on this record in the **SuiteScript Records Browser**. To get the internal IDs for the Shipping sublist, open the Records Browser and navigate to one of the other record types that support this sublist. These records are sales order, cash sale, invoice, and estimate / quote.

The Shipping sublist’s Shipping Tax Code field, internal ID shippingtaxcode, appears only if per-line taxes have been set on the Item sublist.

### Scripting with Item Fulfillment Return Addresses

As of 2018.2, return address fields for the item fulfillment record are replaced by a new return address subrecord. The prior return address fields currently are still supported for SuiteScript, but they are considered to be deprecated. It is recommended that you update any scripts that use these fields to instead use return address subrecord fields.

The internal ID for the new subrecord is returnshippingaddress. The following table maps return address subrecord field IDs to prior return address field IDs.

<table>
<thead>
<tr>
<th><strong>Item Fulfillment Return Address Subrecord ID = returnshippingaddress</strong></th>
<th><strong>Deprecated Return Address Fields</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Label</td>
<td>Field ID</td>
</tr>
<tr>
<td>Country</td>
<td>country</td>
</tr>
<tr>
<td>Attention</td>
<td>attention</td>
</tr>
<tr>
<td>Addressee</td>
<td>addressee</td>
</tr>
<tr>
<td>Phone</td>
<td>addrphone</td>
</tr>
<tr>
<td>Address1</td>
<td>addr1</td>
</tr>
<tr>
<td>Address2</td>
<td>addr2</td>
</tr>
<tr>
<td>City</td>
<td>city</td>
</tr>
<tr>
<td>State</td>
<td>state</td>
</tr>
<tr>
<td>Zip</td>
<td>zip</td>
</tr>
<tr>
<td>Address</td>
<td>addrtext</td>
</tr>
<tr>
<td>Override</td>
<td>override</td>
</tr>
</tbody>
</table>

The following sample shows how to create an item fulfillment return address using the subrecord.

```javascript
require(["N/record"], function(record) {
    var salesOrder = record.transform(
        fromType: record.Type.SALES_ORDER,
        fromId: 1020,
        toType: record.Type.ITEM_FULFILLMENT,
        isDynamic: true,
    );

    var returnAddressSubrecord = salesOrder.getSubrecord(
```
Item Receipt

**Note:** This topic applies to all versions of SuiteScript.

An item receipt transaction records the receipt of returned items from customers. This transaction updates the following information:

- Items on return authorizations are recorded as received.
Inventory records are updated for the new stock levels.
Inventory asset accounts are updated with the values of returned items.
Status of the return is updated.

The item receipt transaction is available when the Advanced Receiving feature is enabled.

For help working with this record in the UI, see the help topics Receiving a Customer Return and Handling Returned Items.

The internal ID of this record is itemreceipt.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The item receipt record is scriptable in both client and server SuiteScript.

**Supported Functions**

The item receipt record is partially scriptable — it can be updated, deleted, and searched using SuiteScript. It cannot be created or copied.

**Usage Notes**

The Item Fulfillment/Item Receipt sublist is a list sublist. In the NetSuite Help Center, see the help topic List Sublists for information on this sublist type.

**Using Landed Cost Fields**

When you create a landed cost category, the associated field IDs for the first category are landedcostamount1 and landedcostsource1. If you create a second category, the IDs will be landedcostamount2 and landedcostsource2.

This pattern increments by one with each additional category. For example, the IDs for the next landed cost category will be landedcostamount3 and landedcostsource3, and so on.

**Creating Item Receipt Records**

You cannot create standalone item receipts using SuiteScript. For example, the following will throw an error:

```javascript
var itemReceipt = record.create({
    type: record.Type.ITEM_RECEIPT
});
```
To create an item receipt, you must use `record.transform(options)`, which transforms the data from one record type, purchase order, for example, into an item receipt. To create an item receipt, your code would be similar to the following:

```javascript
// Transform a record with a specific id to a different record type.
// For example, from PO to Item Receipt
// Get the object of the transformed record.
var trecord = record.transform({
    fromType: record.Type.PURCHASE_ORDER,
    fromId: 26,
    toType: record.Type.ITEM_RECEIPT
});

qty = trecord.getSublistValue({
    sublistId: 'item',
    fieldId: 'quantity',
    line: 1
});

// Set the value of a sublist field.
var trecord = record.transform({
    fromType: record.Type.PURCHASE_ORDER,
    fromId: 26,
    toType: record.Type.ITEM_RECEIPT
});

// Set the value of a sublist field.
var trecord = record.transform({
    fromType: record.Type.PURCHASE_ORDER,
    fromId: 26,
    toType: record.Type.ITEM_RECEIPT
});

var idl = trecord.save ({
    enableSourceing: true
});

email.send({
    author: -5,
    recipient: -5,
    subject: 'Transform Email',
    body: 'Original Qty = ' + qty
});
```

**Item Supply Plan**

*i Note:* This topic applies to all versions of SuiteScript.

The internal ID for this record is `itemsupplyplan`.

This record includes the Orders sublist.

For help working with this record in the UI, see the help topics Generating Item Supply Plans and Item Supply Plan Import.

See the SuiteScript Records Browser for all internal IDs associated with this record.

*i Note:* For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types
The item supply plan is scriptable in both client and server SuiteScript.

Supported Functions
The item supply plan record is fully scriptable — it can be created, copied, updated, deleted, and searched using SuiteScript.

Usage Notes
The item, location, and units body fields cannot be changed in update operations.
An item supply plan’s receiptdate cannot be earlier than the orderdate.
The ordercreated field is read-only. It is set to True when an order is generated from an item supply plan.

Code Samples
The following sample creates an item supply plan.

```javascript
function createItemSupplyPlanMinimal() {
    var itemsupplyplan = record.create({
        type: record.Type.ITEM_SUPPLY_PLAN,
        isDynamic: true
    });
    itemsupplyplan.setValue({
        fieldId: 'subsidiary',
        value: 1
    });
    itemsupplyplan.setValue({
        fieldId: 'location',
        value: 1
    });
    itemsupplyplan.setValue({
        fieldId: 'item',
        value: 165
    });
    itemsupplyplan.setValue({
        fieldId: 'memo',
        value: 'memotest'
    });
    itemsupplyplan.setValue({
        fieldId: 'unit',
        value: 3
    });

    itemsupplyplan.selectNewLine({
        sublistId: 'order'
    });
    itemsupplyplan.setCurrentSublistValue({
        sublistId: 'order',
        fieldId: 'orderdate'
    });
}```
The following sample updates an existing item supply plan.

```javascript
function updateItemSupplyPlan()
{
    var itemsupplyplan = record.load({
        type: record.Type.ITEM_SUPPLY_PLAN,
        id: 3
    });
    itemsupplyplan.setValue({
        fieldId: 'memo',
        value: 'memotest2'
    });
    itemsupplyplan.setSublistValue({
        sublistId: 'order',
        fieldId: 'receiptdate',
        line: 4,
        value: '11/3/2012'
    });
    var id = itemsupplyplan.save();
}
```

Journal Entry

**Note:** This topic applies to all versions of SuiteScript.

You use the journal entry record to adjust balances in your ledger accounts without entering posting transactions.
In the UI, you access this record in the UI at Transactions > Financial > Make Journal Entries.

If your account has the Multi-Book Accounting feature enabled, you can also work with book specific journal entry records, which are available in the UI at Transactions > Financial > Make Book Specific Journal Entries. Although they have different entry forms, both book specific and regular intercompany journal entries are the same record type. Within SuiteScript, they are differentiated by the accountingbook field. In other words, a record that has a value set for accountingbook is book specific. Otherwise, the record is a regular intercompany journal entry.

For help working with this record in the UI, see the help topics Making Intercompany Journal Entries and Book-Specific Journal Entries.

The internal ID for this record is journalentry.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The journal entry record is scriptable in both client SuiteScript and server SuiteScript. All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

The journal entry record is fully scriptable — it can be created, copied, updated, deleted, and searched using SuiteScript.

**Warning:** If you update any type of journal entry that has been applied as a payment to an invoice or vendor bill, the relationship between the journal entry and payment is removed and the payment is no longer applied.

**Usage Notes**

With the Advanced Revenue Management feature, you can directly attach a revenue recognition plan to a book specific journal entry or an intercompany journal entry. Before you begin working with advanced revenue management programmatically, see the help topic Setup for Advanced Revenue Management.

The following table lists the scriptable field associated with this record and advanced revenue management.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Internal ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>End Date</td>
<td>Date</td>
<td>enddate</td>
</tr>
<tr>
<td>Revenue Recognition Rule</td>
<td>List/Record</td>
<td>revenuerecognitionrule</td>
</tr>
</tbody>
</table>
For help working with this record in the UI, see the help topic Creating Revenue Elements from Journal Entries

Code Samples

The following sample shows how to create a book-specific journal entry. The record is book-specific because a value has been set for the accountingbook field.

```javascript
var journalEntry = record.create({
    type: record.Type.JOURNAL_ENTRY,
    isDynamic: true
});
journalEntry.setValue({
    fieldId: 'accountingbook',
    value: '2'                     // Setting a value for this field makes the record book specific.
});
journalEntry.setValue({
    fieldId: 'subsidiary',
    value: '4'
});
journalEntry.setValue({
    fieldId: 'trandate',
    value: '5/16/2013'
});
journalEntry.selectNewLine({
    sublistId: 'line'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: '6'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'credit',
    value: '2.00'
});
journalEntry.commitLine({
    sublistId: 'line'
});
```

```javascript
journalEntry.selectNewLine({
    sublistId: 'line'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: '149'
});
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'debit',
    value: '149'
});
```
Landed Cost

**Note:** This topic applies to all versions of SuiteScript.

The landed cost subrecord supports the Landed Cost Allocation per Line functionality, which is part of the Landed Cost feature. Landed costs typically include location-specific expenses such as customs duties and freight fees. The landed cost subrecord can be used in conjunction with several transactions: check, credit card charge, item receipt, and vendor bill. The purpose of the subrecord is to show the landing costs associated with a particular line in the parent transaction's Items sublist.

The subrecord is available only when the Landed Cost feature is enabled at Setup > Company > Setup Tasks > Enable Features, on the Items & Inventory subtab.

For help working with this record in the UI, see the help topic Using Landed Cost Allocation Per Line on Transactions.

The internal ID for this subrecord is `landedcost`.

**Important:** Landed cost is considered a subrecord, not a record. For general details on working with subrecords, see the help topic Working with Subrecords in SuiteScript. For details about this finding information for subrecords in the Record Browser, see the help topic Finding Subrecord Details in the Records Browser.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The landed cost subrecord is scriptable in server SuiteScript only. The user events are not supported.

**Supported Functions**

The landed cost subrecord is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

To script to the landed cost subrecord, both of the following must be true:

- The parent transaction's Landed Cost per Line option is checked.
- The sublist item being referenced has been configured to use the Track Landed Cost option. For details on configuring the item, see the help topic Set Up Item Records for Landed Cost.
The following sample shows how to create a landed cost subrecord and perform other basic tasks.

```javascript
// Creating a landed cost subrecord
var purchaseOrder = record.create({
    type: record.Type.PURCHASE_ORDER,
    isDynamic: true
});
purchaseOrder.setText({
    fieldId: 'entity',
    text: 'Acme Medical Supply'
});
purchaseOrder.setSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    line: 1,
    value: inventoryItemId                 // defined outside this snippet
});

var purchaseOrderId = purchaseOrder.save();

var itemReceipt = record.transform({
    fromtype: record.Type.PURCHASE_ORDER,
    fromId: purchaseOrderId,              // defined outside this snippet
    toType: record.Type.ITEM_RECEIPT
});
itemReceipt.selectLine({
    sublistId: 'item',
    line: 1
});
itemReceipt.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'location',
    value: 1
});
itemReceipt.setValue({
    fieldId: 'landedcostperline',
    value: 'T'
});

var landedCost = itemReceipt.getCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'landedcost'
});
landedCost.selectNewLine({
    sublistId: 'landedcostdata'
});
landedCost.setCurrentSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'costcategory',
    value: 1
});
landedCost.setCurrentSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'scheduledcost'
});
```
fieldId: 'amount',
value: 456
});
landedCost.commitLine({
  sublistId: 'landedcostdata'
});
landedCost.selectNewLine({
  sublistId: 'landedcostdata'
});
landedCost.setCurrentSublistValue({
  sublistId: 'landedcostdata',
  fieldId: 'costcategory',
  value: 3
});
landedCost.setCurrentSublistValue({
  sublistId: 'landedcostdata',
  fieldId: 'amount',
  value: 78.96
});
landedCost.commitLine({
  sublistId: 'landedcostdata'
});
itemReceipt.commitLine({
  sublistId: 'item'
});

var itemReceiptId = itemReceipt(save);

// Viewing the subrecord
itemReceipt = record.load ({
  type: record.Type.ITEM_RECEIPT,
  id: itemReceiptId
});
itemReceipt.selectLine({
  sublistId: 'item',
  line: 1
});
landedCost = itemReceipt.getSublistSubrecord({
  sublistId: 'item',
  fieldId: 'landedcost',
  line: 1
});
landedCost.getSublistValue({
  sublistId: 'landedcostdata',
  fieldId: 'amount',
  value: 1
});

// Updating the subrecord
landedCost = itemReceipt.getCurrentSublistSubrecord({
  sublistId: 'item',
  fieldId: 'landedcost'
});
landedCost.removeLine({
  sublistId: 'landedcostdata',
  value: 1
});
line: 2
});

landedCost.setSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'costcategory',
    line: 1,
    value: 2
});
landedCost.setSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'amount',
    line: 1,
    value: 3.98
});
landedCost.selectNewLine({
    sublistId: 'landedcostdata'
});
landedCost.setCurrentSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'costcategory',
    value: 3
});
landedCost.setCurrentSublistValue({
    sublistId: 'landedcostdata',
    fieldId: 'amount',
    value: 103
});
landedCost.commitLine({
    sublistId: 'landedcostdata'
});
itemReceipt.commitLine({
    sublistId: 'item'
});
var receiptId = itemReceipt.save();

// Deleting the subrecord
itemReceipt.selectLine({
    sublistId: 'item',
    line: 1
});
itemReceipt.removeCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'landedcost'
});
itemReceipt.commitLine({
    sublistId: 'item'
});
var receiptId2 = itemReceipt.save();
Manufacturing Operation Task

**Note:** This topic applies to all versions of SuiteScript.

The Manufacturing Routing and Work Center feature lets you specify a sequence of tasks required for the completion of a Work In Process (WIP) work order. This record represents a job that must be completed by a specific employee group. In the UI, generally these records are created automatically when you save a WIP work order that references a specific routing record — each step described in the routing record becomes an operation task record, viewable on the work order’s Operations subtab. You can also manually create an operation task record by clicking the New Operation Task button on the work order’s Operations subtab. You can view all existing manufacturing operation task records by going to Transactions > Manufacturing > Manufacturing Operation Tasks.

For help working with this record in the UI, see the help topic Manufacturing Operation Tasks.

The internal ID for this record is **manufacturingoperationtask**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The manufacturing operation task record is scriptable in both client and server SuiteScript.

**Supported Functions**

The manufacturing operation task record is partially scriptable — it can be created, updated, deleted, and searched using SuiteScript. It cannot be copied.

**Usage Notes**

To work with the manufacturing operation task record, the Manufacturing Routing and Work Center feature must be enabled at Setup > Company > Enable Features, on the Items & Inventory tab.

In the UI, the manufacturing operation task record is accessed by going to Transactions > Manufacturing > Manufacturing Operation Tasks. Alternatively, you can navigate to the Operations subtab of a WIP work order that uses the routing feature. The Operations subtab lists existing operation task records for that work order and allows you to create new operation task records.

Note these additional details:

- This record supports client and server scripting.
All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

To create a new manufacturing operation task record, you must reference a valid existing WIP work order, as shown in the Code Samples below.

Code Samples

When creating a manufacturing operation task record, you must use initValues to reference a valid existing WIP work order. For example:

```javascript
var initValues = new Array();
initValues.workorder = '65';
var task = record.create({
    type: record.Type.MANUFACTURING_OPERATION_TASK,
    defaultValues: initValues
});
task.setValue({
    fieldId: 'title',
    value: 'Some title'
});
task.setValue({
    fieldId: 'operationsequence',
    value: 6
});
task.setValue({
    fieldId: 'setuptime',
    value: 30
});
task.setValue({
    fieldId: 'runrate',
    value: 20
});
task.setValue({
    fieldId: 'manufacturingcosttemplate',
    value: '1'
});
task.setValue({
    fieldId: 'manufacturingworkcenter',
    value: '113'
});
var recId = task.save();
```

Manufacturing Planned Time

Note: This topic applies to all versions of SuiteScript.

The Manufacturing Planned Time search type enables you to search for data displayed on the Planned Time subtab of work orders. This subtab is available only when the Show Planned Capacity on Work Orders preference is enabled.

The Planned Time subtab is used to display data about the amount of time being allocated to each work center per day for the work order. This data is derived from the associated operation task records.
You can set the Show Planned Capacity on Work Orders preference at Setup > Accounting > Accounting Preferences. The preference is available only when the Manufacturing Routing and Work Center feature is enabled.

For help working with this record in the UI, see the help topic Manufacturing Routing.

The internal ID for this record is mfgplannedtime.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This search is supported in both client and server SuiteScript.

**Supported Functions**

Only search is supported for this record.

**Usage Notes**

Be aware of the following:

- This record is a search record only. You cannot create or copy this record.
- In the UI, you can view this search by navigating to Reports > New Search and clicking Manufacturing Planned Time.

For help creating search scripts, see the help topic Search Samples.

**Memorized Transaction Definition**

**Note:** This topic applies to all versions of SuiteScript.

A memorized transaction is a transaction you set up to recur in NetSuite, such as recurring journal entries. Memorized transactions eliminate data entry and serve as useful reminders. A memorized transaction consists of two parts:

- The Memorized Transaction Template defines the transaction to be created.
- The Memorized Transaction Definition sets the parameters for when to create the transaction, whether they are recurring, and how many transactions to create.

Memorized transaction templates and definitions must be created in the NetSuite user interface, but existing memorized transaction definitions can be called using SuiteScript.
When you memorize a transaction, you set up the transaction to recur over time. You choose whether the transaction automatically posts or if you are reminded to post it. You also set the frequency the transaction should occur and set posting options. You can specify that the transaction should recur forever or a set number of times.

When you edit or delete a memorized transaction, the change is tracked on the system notes of the memorized transaction.

The following are limitations for posting memorized transactions in locked and closed periods:

- A user must have the Override Period Restrictions permission to enable the Allow Posting in Locked Period option for a memorized transaction. When this option is enabled, memorized transactions can post in locked periods even if users initiating these transactions do not have the Override Period Restrictions permission.
- Transactions can never be posted in closed periods. If the posting date for a memorized transaction is in a period that has been closed, it is posted to the next open period.

In NetSuite, you can memorize the following transactions:

- Checks
- Credit Cards
- Purchase Orders
- Bills
- Sales Orders
- Invoices
- Estimates
- Cash Sales
- Journal Entries
- Statement Charges
- Work Orders
- Purchase Requisitions
- Transfer Orders
- Opportunities
- Custom Transactions

For help working with this record in the UI, see the following articles:

- Memorized Transactions
- Memorizing a Transaction
- Editing Memorized Transactions
- Customizing the Memorized Transactions List
- Viewing the Status of Memorized Transactions
- Deleting Memorized Transactions

**Supported Script Types**

Memorized transaction definitions are supported in client and server SuiteScript.
**Note:** Automatically-created transactions do not support server-side scripting.

**Supported Functions**

Memorized transaction definitions support only the Read, Update, Delete, and Search functions.

**Usage Notes**

Be aware of the following:

- The System Notes and Transaction subtabs have not been exposed and cannot be used in scripts.
- Transforming memorized transaction definition records into other types of records is not supported.
- This record has one subrecord: a list with record ID “customdates”.
- You cannot call nlapiCreateRecord with client-side scripts using memorized transaction definitions.
- Scripts which run when a transaction is created may not work as expected if the transaction is automatically created from a memorized transaction. For example, you may have a script which runs when you save a purchase order. However, the script will not run if the purchase order was created from a memorized transaction definition which has “Automatic” set in its Action field.

For information about scripting with this record in SuiteScript, see the following help topics:

- Working with the SuiteScript Records Browser

**Sample Code**

Memorized transaction definitions can only be created in the NetSuite user interface. The example script below first loads an existing memorized transaction definition with ID 2. Then, it sets the Action field to “Reminder”, and sets the Next Transaction Date in the Custom Dates subtab to 12/12/2020. It saves this date, and then saves the memorized transaction definition.

```javascript
var r = nlapiLoadRecord('memdoc', 2);
r.setFieldValue('action', 'REMINDER');
r.selectNewLineItem('customdates');
r.setCurrentLineItemValue('customdates', 'nextdate', '12/12/2020');
r.commitLineItem('customdates');
var id = nlapiSubmitRecord(r);```

**Multi-Book Accounting Transaction**

**Note:** This topic applies to all versions of SuiteScript.

If your account has the Multi-Book Accounting feature enabled, you can use SuiteScript to search for transactions using accounting book as a search filter or a search column. To execute this type of search, you use the multi-book accounting transaction search record.

In the UI, you can view the multi-book accounting transaction search by navigating to Reports > New Search and clicking Multi-Book Accounting Transaction.
For help working with this record in the UI, see the help topic Multi-Book Accounting.

The internal ID for this record is `accountingtransaction`. Note that this record is a search record only. You cannot create or copy this record.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The multi-book accounting transaction record is supported in both client and server SuiteScript.

**Supported Functions**

Only search is supported for this record.

**Sample Code**

The following sample shows how to search for a specific transaction and include in the results details about the transaction's accounting book and foreign exchange rate, among other data.

```javascript
var mySearch = search.create (
  { type: search.Type.TRANSACTION,
    columns: [{
      name: 'accountingbook',
      sort: search.Sort.ASC
    }, {
      name: 'line',
      join: 'transaction',
      sort: search.Sort.ASC
    }, {
      name: 'account'
    }, {
      name: 'amount'
    }, {
      name: 'exchangerate'
    }],
    filters: [{
      name: 'internalid',
      operator: search.Operator.IS,
      values: 18                      // last parameter represents the actual internal ID of the transaction
    }],
  });
```
for (var i = 0; mySearch != null && i < mySearch.length; i++) {
  var searchresult = mySearch[i];
  var record = searchresult.getId();
  var rectype = searchresult.getRecordType();
  var book = searchresult.getValue({
    name: 'accountingbook'
  });
  var line = searchresult.getValue({
    name: 'line',
    join: 'transaction'
  });
  var account = searchresult.getValue({
    name: 'account'
  });
  var amount = searchresult.getValue({
    name: 'amount'
  });
  var fxrate = searchresult.getValue({
    name: 'exchangerate'
  });

  alert(i + ': ' + book + ' | ' + line + ' | ' + record + ' | ' + rectype + ' | ' + account + ' | ' + amount + ' | ' + fxrate);
}

Opportunity

Note: This topic applies to all versions of SuiteScript.

Opportunities represent negotiations with prospects. You must first enable opportunities in your NetSuite account before you can access this record type. To enable opportunities, go to Setup > Company > Enable Features. On the CRM tab, under Sales, select the Opportunities check box.

For help working with this record in the UI, see Opportunity Records.

The internal ID for this record is opportunity.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The opportunity record is scriptable in both client and server SuiteScript.
Supported Functions

The manufacturing operation task record is fully scriptable — it can be created, copied, updated, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrosprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrosprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

Also note that this record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

Order Schedule

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Commit Order Scheduling.

The internal ID for this subrecord is `orderschedule`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types

Server scripts for this subrecord must access through the parent record.

Supported Functions

For details, see the help topic Working with Subrecords in SuiteScript.

Usage Notes

Order Schedule is scriptable from both the body field and the line item.
Order Schedule is considered a subrecord in SuiteScript.

Paycheck

**Note:** This topic applies to all versions of SuiteScript.

This record is available only when the Payroll feature is enabled at Setup > Company > Enable Features on the Employees subtab.

In the UI, you can access this record at Transactions > Employees > Create Payroll.

For help working with this record in the UI, see the help topics Creating a Payroll Batch and Editing an Individual Paycheck from a Payroll Batch.

The internal ID for this record is `paycheck`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

This record is scriptable in both client and server SuiteScript.

Supported Functions

The paycheck record is partially scriptable — it can be read and updated only.
Usage Notes

Be aware of the following:

- The paycheck record is created through the payroll batch record.

Code Samples

The following sample shows how to search for paycheck records.

```javascript
var mySearch = search.create({
    type: search.Type.PAYCHECK,
    columns: [{
        name: 'batchnumber',
    }, {
        name: 'employee',
    }],
    filters: [{
        name: 'batchnumber',
        operator: search.Operator.EQUALTO,
        values: 100
    }],
});
```

Paycheck Journal

**Note:** This topic applies to all versions of SuiteScript.

The Paycheck Journal record is intended to enable global payroll solutions. You can use it along with the Payroll Item record to create custom payroll solutions and to support integrations with external payroll systems.

For help working with this record in the UI, see the help topic **Using the Paycheck Journal Feature**.

The internal ID for this record is **paycheckjournal**.

See the [SuiteScript Records Browser](https://www.netsuite.com) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic **Working with the SuiteScript Records Browser** in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Usage Notes

The Paycheck Journal feature must be enabled to work with the Paycheck Journal record.
The following sample creates a paycheck journal transaction.

```javascript
// create Paycheck Journal with two earnings and one deduction sublist
function createPaycheckJournal(){
    var pj = record.create ({
        type: record.Type.PAYCHECK_JOURNAL,
        isDynamic: true
    });
    pj.setValue({
        fieldId: 'trandate',
        value: '6/10/2012'
    });
    pj.setValue({
        fieldId: 'employee',
        value: 4  // internal ID of the employee
    });
    pj.setValue({
        fieldId: 'account',
        value: 28  // internal ID of the account
    });
    pj.selectNewLine({
        sublistId: 'earning'
    });
    pj.setCurrentSublistValue({
        sublistId: 'earning',
        fieldId: 'payrollitem',
        value: '102'
    });
    pj.setCurrentSublistValue({
        sublistId: 'earning',
        fieldId: 'amount',
        value: 20.35
    });
    pj.commitLine({
        sublistId: 'earning'
    });
    pj.selectNewLine({
        sublistId: 'earning'
    });
    pj.setCurrentSublistValue({
        sublistId: 'earning',
        fieldId: 'payrollitem',
        value: '102'
    });
    pj.setCurrentSublistValue({
        sublistId: 'earning',
        fieldId: 'amount',
        value: 33.35
    });
    pj.commitLine({
        sublistId: 'earning'
    });
}
```
The following sample updates a paycheck journal transaction.

```javascript
// update Paycheck journal by setting new amount of line 2 earning list
// and clearing the deduction list
function updatePaycheckJournal(){
    var pj = record.load({
        type: record.Type.PAYCHECK_JOURNAL,
        isDynamic: true,
        id: 305
    });

    pj.setSublistValue({
        sublistId: 'earning',
        fieldId: 'amount',
        line: 2, // 1 is the line number we intend to update
        value: 444.44
    });

    var lineCount = pj.getLineCount({
        sublistId: 'deduction'
    });

    for (var lineNo = 1; lineNo <= lineCount; lineNo++) {
        pj.removeLine({
            sublistId: 'deduction',
            line: lineNo
        });
    }

    var recId = pj.save();
}
```
Payroll Batch

**Note:** This topic applies to all versions of SuiteScript.

This record is only available for Payroll customers. For more information, see the help topic [SuitePeople Overview](#).

In the UI, you can access this record at Transactions > Employees > Create Payroll.

For help working with this record in the UI, see the help topic [Creating a Payroll Batch](#).

The internal ID of this record is `payrollbatch`.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The payroll batch record is scriptable in both client, and server SuiteScript.

**Supported Functions**

The payroll batch record is fully scriptable, which means that the record can be created, updated, copied, deleted and searched using SuiteScript.

**Usage Notes**

Be aware of the following:

- After a payroll batch is created it is submitted twice. One time to calculate and again to commit.
- To calculate a payroll batch, it needs to be reloaded after it is submitted until one of the following status' is generated: calculated or error, committed or error.
- If you are calculating and committing a large amount of data a loop should be written to recheck the status of the payroll batch. Refer to the following table for more details.

<table>
<thead>
<tr>
<th>Payroll Batch Status</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Created</td>
<td>A</td>
<td>A new payroll batch has been created.</td>
</tr>
<tr>
<td>Calculated</td>
<td>B</td>
<td>A payroll batch has been calculated.</td>
</tr>
<tr>
<td>Edited</td>
<td>C</td>
<td>A payroll batch may have been calculated previously, or it is a newly created batch with some paychecks added or calculated.</td>
</tr>
<tr>
<td>About to Commit</td>
<td>D</td>
<td>Payroll is about to be committed.</td>
</tr>
<tr>
<td>Committed At Service</td>
<td>E</td>
<td>Payroll is committed for processing by payroll service, but some records need to be created in NetSuite.</td>
</tr>
<tr>
<td>Committed</td>
<td>F</td>
<td>Payroll is committed.</td>
</tr>
</tbody>
</table>
## Payroll Batch Status

<table>
<thead>
<tr>
<th>Status</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed</td>
<td>P</td>
<td>Payroll batch is completed.</td>
</tr>
<tr>
<td>Reversed</td>
<td>R</td>
<td>Payroll batch is reversed.</td>
</tr>
<tr>
<td>Error</td>
<td>X</td>
<td>Error occurred in payroll batch.</td>
</tr>
</tbody>
</table>

## Code Samples

The following sample shows how to create a payroll batch and perform other basic tasks. It also includes how to add an employee to a payroll batch. For more information, see Payroll Batch Employee.

```javascript
//Create the payroll batch record
var payrollbatch = record.create({
    type: 'payrollbatch',
    isDynamic: true
});
payrollbatch.setValue({
    fieldId: 'offcycle',
    value: 'F'
});
payrollbatch.setValue({
    fieldId: 'payfrequency',
    value: '52'
});
payrollbatch.setValue({
    fieldId: 'periodending',
    value: '6/15/2016'
});
var recId = payrollbatch.save();

// Add employees to the payroll batch
// In most cases, you would iterate through the addmorepayeesmachine to find the employees you would add to the batch
var addemp = payrollbatch.load({
    type: 'payrollbatchaddemployees',
    id: id,
    defaultValues: 'true'
});
addemp.setSublistValue({
    sublistId: 'addmorepayeesmachine',
    fieldId: 'payemp',
    line: 1,
    value: 'T'
});
var recId2 = addemp.save();

//Submit the record to be calculated
payrollbatch = record.load({
    type: 'payrollbatch',
    id: id,
    defaultValues: 'false'
});
```
```javascript
var recId3 = payrollbatch.save();

// Check the status of the record to ensure that calculating is complete before proceeding
// Do { 
    payrollbatch = record.load({
        type: 'payrollbatch',
        id: id,
        defaultValues: 'false'
    });

    status = payrollbatch.getValue({
        fieldId: 'status'
    });
} while ( !(status == 'B' || status == 'E' || status == 'X') );

// Set the flag to commit the payroll batch
if (status == 'B') {
    payrollbatch.setValue({
        fieldId: 'commit',
        value: 'T'
    });
}

// Commit
var recId3 = payrollbatch.save();
```

```javascript
// Do { 
    payrollbatch = record.load({
        type: 'payrollbatch',
        id: id,
        defaultValues: 'false'
    });

    status = payrollbatch.getValue({
        fieldId: 'status'
    });
} while ( !(status == 'F' || status == 'E' || status == 'X') );
```

---

**Payroll Batch Employee**

**Note:** This topic applies to all versions of SuiteScript.

This record enables you to add employees to the payroll batch. This record is only available for Payroll customers. For more information, see the help topic [SuitePeople Overview](#).

In the UI, you can access this record at Transactions > Employees > Create Payroll.

For help working with this record in the UI, see the help topic [Adding Employees to a Payroll Batch](#).

The internal ID of this record is `payrollbatchaddemployees`.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
Supported Script Types

This record is scriptable in both client and server SuiteScript.

Supported Functions

The payroll batch employee record is read and edit only.

Usage Notes

Be aware of the following:

- You cannot deselect an employee through SuiteScript after they have been added to the payroll batch and the record is submitted. If you need to remove an employee from the payroll batch you will need to do this through the UI.

Price Plan

Note: This topic applies to all versions of SuiteScript.

The price plan record, along with the price book record, is used to build pricing information for subscriptions.

The price plan record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features (Administrator), on the Transactions subtab.

For help working with this record in the UI, see the help topic Creating Price Plans.

The internal ID for this record is priceplan.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The price plan record is scriptable in server and client SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Supported Functions

The price plan record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Period End Journal

**Note:** This topic applies to all versions of SuiteScript.

Period End Journal is a transaction record that posts subsidiary consolidation and year end closing to the general ledger. Period end journals are generated automatically when you complete the Create Period End Journals task on the Period Close Checklist.

For help working with this record in the UI, see the help topic [Period End Journal Entries](#).

This record is part of the Period End Journal Entries feature, which is available only in NetSuite OneWorld accounts. The Accounting Periods feature is a prerequisite for the Period End Journal Entries feature. Before you begin working with period end journals programmatically, see the help topic [Setup for Period End Journal Entries](#).

The internal ID for this record is `periodendjournal`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#).

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The period end journal record is scriptable in server and client SuiteScript. The system process that generates this record does not trigger user events. However, user events can be triggered outside of the process.

**Supported Functions**

The period end journal record can be read, updated, and searched using SuiteScript. However, update is limited to the Memo fields and any custom fields or custom segments that do not affect the general ledger. When searching for period end journals, Include Period End Transactions must be set to True in the results.

**Code Samples**

The following sample shows how to update the value of the memo field.

```javascript
var periodEndJournal = record.load({
    type: record.Type.PERIOD_END_JOURNAL,
    id: 101,
    isDynamic: true
});
periodEndJournal.setText({
    fieldId: 'memo',
    text: 'Test Memo'
});
periodEndJournal.save();
```
This record enables you to take advantage of contracted quantity-based terms and discounts when creating purchase orders.

This record is available only when the Purchase Contracts feature is enabled at Setup > Company > Enable Features, on the Transactions subtab.

In the UI, you access this record at Transactions > Purchases > Enter Purchase Contracts.

For help working with this record in the UI, see the help topic Creating Purchase Contracts.

The internal ID of this record is **purchasecontract**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

**Supported Script Types**

This record is scriptable in both client and server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

This record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

**Purchase Order**

A purchase order transaction authorizes the purchase of goods and/or services. This transaction tracks items expected to be received, items actually received, and items yet to be received. A purchase order
has no accounting impact until the included items are received. This transaction is enabled when the Purchase Orders feature is enabled.

For help working with this record in the UI, see the help topic Purchasing.

The internal ID for this record is `purchaseorder`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The purchase order record is scriptable in both client and server SuiteScript.

**Supported Functions**

This record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

**Usage Notes**

See the following sections for more details on this record:
- Fields
- Transforms
- Email Operations

**Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit&lt;br&gt;When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.&lt;br&gt;When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent&lt;br&gt;When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.&lt;br&gt;When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost&lt;br&gt;When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>
Transforms

This record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

Email Operations

NetSuite enables a user to email a copy of a transaction to a customer or another recipient.

If you deploy a `beforeLoad` user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction. However, if the companywide preference `Use Popup for Main Transaction Email Button` is enabled, a beforeLoad script will not execute in certain cases, depending on how the email is sent. That is, the script will not execute if the user views the transaction and selects Actions > Email, or edits the transaction and sends the email by using the Communications tab. In these cases, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this window are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the `Message` record type.

An additional way of emailing a transaction is by editing the transaction and using the Save & Email button. Note that, even if the Use Popup for Main Transaction Email Button preference is enabled, the popup window is not available for emails sent this way. For that reason, a beforeLoad user event script deployed on the transaction will always be triggered by use of the Save & Email button.

For more information on the Use Popup for Main Transaction Email Button preference, see the help topic Setting Email Preferences. For details about emailing transactions, see the help topic Emailing Transactions.

Requisition

**Note:** This topic applies to all versions of SuiteScript.

You use the requisition record to initiate the purchase process for goods and services needed within your company.

This record is available only when the Requisitions feature is enabled, at Setup > Enable Features, on the Transactions subtab. In the UI, you access this record at Transactions > Purchases/Vendors > Enter Requisitions.

For help working with this record in the UI, see the help topic Entering a Requisition.

The internal ID for this record is `purchaserequisition`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
N/record Module

Supported Script Types
The requisition record is scriptable in both client and server SuiteScript.
All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Supported Functions
The requisition record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Code Samples
The following sample shows how to add a requisition record and perform other basic tasks.

```javascript
// Requisition SuiteScript name
var recType = record.Type.PURCHASE_REQUISITION;

// Create and Add Record
var purchaseReq = record.create({
  type: recType,
  isDynamic: true
});
purchaseReq.setValue({
  fieldId: 'location',
  value: 1
});
purchaseReq.setSublistValue({
  sublistId: 'expense',
  fieldId: 'account',
  line: 1,
  value: 59
});
purchaseReq.setSublistValue({
  sublistId: 'expense',
  fieldId: 'amount',
  line: 1,
  value: 2.3
});
var recId = purchaseReq.save();

// Load and Update
purchaseReq = record.load({
  type: recType,
  id: recId
});
purchaseReq.setValue({
  fieldId: 'memo',
  value: 'memo'
});
var recId2 = purchaseReq.save();
```
var result = search.create({
    type: search.Type.TRANSACTION,
    filters: [{
        name: 'internalId',
        operator: search.Operator.ANYOF,
        values: recId
    }],
    columns: [{
        name: 'memo'
    }]
});

var purchaseReq2 = record.copy({
    type: recType,
    id: recId
});
var rec2Id = purchaseReq2.save();

var purchaseOrder = record.transform({
    fromType: recType,
    fromId: rec2Id,
    toType: record.Type.PURCHASE_ORDER
});
purchaseOrder.setValue({
    fieldId: 'entity',
    value: '105'
});
var poId = purchaseOrder.save();

var delRecId1 = record.delete({
    type: record.Type.PURCHASE_ORDER,
    id: poId
});
var delRecId2 = record.delete({
    type: recType,
    id: recId
});
var delRecId3 = record.delete({
    type: recType,
    id: rec2Id
});

---

Return Authorization

**Note:** This topic applies to all versions of SuiteScript.

A return authorization transaction, also known as a return materials authorization (RMA), records information about an expected return of items from a customer, including the item IDs, vendors,
quantities, and prices (which determine the amounts to be credited or refunded to the customer. This
transaction is non-posting. It is available when the Return Authorizations feature is enabled.

For help working with this record in the UI, see the help topic Customer Return Management.
The internal ID for this record is `returnauthorization`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with
the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**
The return authorization record is scriptable in both client and server SuiteScript.

**Supported Functions**
The return authorization record is fully scriptable, which means that the record can be created, updated,
copied, deleted, and searched using SuiteScript. It can also be transformed.

**Usage Notes**
See the following sections for more details on working with this record:
- Transforms
- Email Operations

**Fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
</tr>
</tbody>
</table>

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to
appear.
Transforms

This record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

Email Operations

NetSuite enables a user to email a copy of a transaction to a customer or another recipient.

If you deploy a beforeLoad user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction. However, if the companywide preference Use Popup for Main Transaction Email Button is enabled, a beforeLoad script will not execute in certain cases, depending on how the email is sent. That is, the script will not execute if the user views the transaction and selects Actions > Email, or edits the transaction and sends the email by using the Communications tab. In these cases, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this window are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the Message record type.

An additional way of emailing a transaction is by editing the transaction and using the Save & Email button. Note that, even if the Use Popup for Main Transaction Email Button preference is enabled, the popup window is not available for emails sent this way. For that reason, a beforeLoad user event script deployed on the transaction will always be triggered by use of the Save & Email button.

For more information on the Use Popup for Main Transaction Email Button preference, see the help topic Setting Email Preferences. For details about emailing transactions, see the help topic Emailing Transactions.

Revenue Arrangement

Revenue Arrangement is a transaction record that contains the details of customer performance obligations for purposes of revenue allocation and recognition.

This record is part of advanced revenue management. To use advanced revenue management, the Accounting Periods feature and Advanced Revenue Management feature must be enabled. Before you begin working with advanced revenue management programmatically, see the help topic Setup for Advanced Revenue Management.

For help working with this record in the UI, see the help topic Revenue Arrangement Management.

The internal ID for this record is `revenuearrangement`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

See Revenue Arrangement Record Actions for actions associated with this record. For more information about actions and macros, see the help topic Overview of Record Action and Macro APIs.

For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
### Supported Script Types

The revenue arrangement record is scriptable in server SuiteScript only.

### Supported Functions

The return authorization record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

### Revenue Element Sublist

The internal id for this sublist is `revenueelement`. The sublist type is inline editor.

The following table lists the scriptable line item fields on the Revenue Element sublist. Revenue elements correspond to individual lines in a source transaction.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Internal ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allocation Type</td>
<td>List/Record</td>
<td>allocationtype</td>
</tr>
<tr>
<td>Amortization End Date</td>
<td>Date</td>
<td>amortizationenddate</td>
</tr>
<tr>
<td>Amortization Start Date</td>
<td>Date</td>
<td>amortizationstartdate</td>
</tr>
<tr>
<td>Base Fair Value</td>
<td>Decimal Number</td>
<td>fairvalue</td>
</tr>
<tr>
<td>Calculated Fair Value Amount</td>
<td>Decimal Number</td>
<td>calculatedamount</td>
</tr>
<tr>
<td>Contract Expense Account</td>
<td>List/Record</td>
<td>contractexpenseacct</td>
</tr>
<tr>
<td>Contract Expense Offset Account</td>
<td>List/Record</td>
<td>contractexpenseoffsetacct</td>
</tr>
<tr>
<td>Cost Amortization Amount</td>
<td>Decimal Number</td>
<td>costamortizationamount</td>
</tr>
<tr>
<td>Deferral Account</td>
<td>List/Record</td>
<td>deferralaccount</td>
</tr>
<tr>
<td>End Date</td>
<td>Date</td>
<td>revrecenddate</td>
</tr>
<tr>
<td>Fair Value Override</td>
<td>Check Box</td>
<td>fairvalueoverride</td>
</tr>
<tr>
<td>VSOE</td>
<td>Check Box</td>
<td>isvsoeprice</td>
</tr>
<tr>
<td>Permit Discount</td>
<td>Check Box</td>
<td>permitdiscount</td>
</tr>
<tr>
<td>Reference ID</td>
<td>Long Text</td>
<td>referenceid</td>
</tr>
<tr>
<td>Recognition Account</td>
<td>List/Record</td>
<td>recognitionaccount</td>
</tr>
<tr>
<td>Return of Element</td>
<td>List/Record</td>
<td>returnofelement</td>
</tr>
<tr>
<td>Revenue Allocation Group</td>
<td>List/Record</td>
<td>revenueallocationgroup</td>
</tr>
<tr>
<td>Revenue Amount</td>
<td>Decimal Number</td>
<td>allocationamount</td>
</tr>
</tbody>
</table>
**Revenue Arrangement Record Actions**

The Revenue Arrangement record currently supports the allocate action.

For help working with this record in the UI, see Revenue Arrangement.

For information about SuiteScript 2.0 record actions, see the following help topics:

- Overview of Record Action and Macro APIs
- N/action Module

### allocate

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Allocate</th>
</tr>
</thead>
</table>
| **Action Description**  | Redistributes a transaction amount across revenue elements in a specified revenue arrangement, based on fair value, to obtain the revenue amount. This process is required for revenue arrangements that are not compliant, meaning the amount has not been allocated.  
Allocation is usually automatic. However, a revenue arrangement may not be compliant because allocation has failed, or the record has been edited to set the Compliant value to false.
For more information, see the help topics Revenue Reallocation for Revenue Arrangements and Fair Value and Allocation. |
| **Returns**             | {notifications:[], response:{}} |
| **Supported Script Types** | Client and server scripts  
For additional information, see the help topic SuiteScript 2.0 Script Types. |
| **Governance**          | 10 usage units |
| **Since**               | 2018.2 |

### Parameters

See the help topic `action.execute(options)` for details about parameters required for the execution of any action. The allocate action does not support any additional parameters.

### Revenue Arrangement Allocate Action Syntax

```javascript
require(['N/action'], function(action) {
    // action & action list loading
    var actionList = action.find({recordType: 'revenuearrangement'});
    var actionObj = action.get({recordType: 'revenuearrangement', id: 'allocate'});

    // action execution
    var result = actionObj.execute({recordId: 2});
    var result = actionObj.execute({recordId: 2});
    var result = action.execute({recordType: 'revenuearrangement', id: 'allocate', params: {recordId: 2}});
});
```
Revenue Commitment

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Using Revenue Commitments. The internal ID for this record is `revenuecommitment`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The revenue commitment record is scriptable in client and server SuiteScript.

**Supported Functions**

The revenue commitment record is partially scriptable. It can be updated, copied, deleted, and searched using SuiteScript. It cannot be created.

**Usage Notes**

You cannot create this record using the standard `record.create(options)` function. To create a Revenue Commitment record, you must execute a Sales Order to Revenue Commitment transformation. In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to appear.

**Revenue Commitment Reversal**

**Note:** This topic applies to all versions of SuiteScript.

If a revenue commitment cannot be invoiced, or the revenue and unbilled receivable must be reversed for some reason, a user with sufficient permission must reverse the revenue commitment.

For help working with this record in the UI, see the help topic Creating Revenue Commitment Reversals.
The internal ID for this record is `revenuecommitmentreversal`.
See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The revenue commitment reversal record is scriptable in client and server SuiteScript.

### Supported Functions

The revenue commitment reversal record is partially scriptable. It can be updated, copied, deleted, and searched using SuiteScript. It cannot be created.

### Usage Notes

You cannot create this record using the standard `record.create(options)` function. To create a Revenue Commitment Reversal record, you must execute a Return Authorization to Revenue Commitment Reversal transformation. Note that the Return Authorization must be approved and received for the transform to work.

In the NetSuite Help Center, see the help topic `record.transform(options)` for examples on how to transform records.

### Sales Order

**Note:** This topic applies to all versions of SuiteScript.

A sales order transaction records a commitment to sell goods or services to a customer. Sales orders have no accounting impact until items are shipped or services are completed.

For help working with this record in the UI, see the help topic Sales Orders.

The internal ID for this record is `salesorder`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

See the Sales Order Record Macro for the macro associated with this record. For more information about actions and macros, see the help topic Overview of Record Action and Macro APIs.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types
The sales order record is scriptable in both client and server SuiteScript.

Supported Functions
The return authorization record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript. It can also be transformed.

Usage Notes
See the following sections for more details about working with this record:
- Fields
- Email Operations
- Transforming Sales Orders with the Intercompany Cross-Subsidiary Feature

Fields
The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrosprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrosprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>isrecurringpayment</td>
<td>Recurring Payment</td>
<td>A value for this field is stored only if the value for paymentmethod is a credit card.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

Also note that this record has available transforms. See the SuiteScript Records Browser for available transforms. In the NetSuite Help Center, see the help topic record.transform(options) for examples on how to transform records.

The Shipping sublist's Shipping Tax Code field, internal ID shippingtaxcode, appears only if per-line taxes have been set on the Item sublist.
Email Operations

NetSuite enables a user to email a copy of a transaction to a customer or another recipient.

If you deploy a beforeLoad user event script on a transaction, in general the script executes when the transaction is emailed. In these cases, the sending of the email is considered an operation of type email on the transaction. However, if the companywide preference **Use Popup for Main Transaction Email Button** is enabled, a beforeLoad script will not execute in certain cases, depending on how the email is sent. That is, the script will not execute if the user views the transaction and selects Actions > Email, or edits the transaction and sends the email by using the Communications tab. In these cases, the system displays a popup window that the user can work with to manually edit and send the email. Actions taken with this window are not considered operations on the transaction. If you have your system configured this way, and you want to deploy a beforeLoad script on emails sent using the popup, deploy the script on the **Message** record type.

An additional way of emailing a transaction is by editing the transaction and using the Save & Email button. Note that, even if the Use Popup for Main Transaction Email Button preference is enabled, the popup window is not available for emails sent this way. For that reason, a beforeLoad user event script deployed on the transaction will always be triggered by use of the Save & Email button.

For more information on the Use Popup for Main Transaction Email Button preference, see the help topic **Setting Email Preferences**. For details about emailing transactions, see the help topic **Emailing Transactions**.

Transforming Sales Orders with the Intercompany Cross-Subsidiary Feature

If the Intercompany Cross-Subsidiary Fulfillment feature is enabled if your account, inventory location may become a mandatory transformation parameter when transforming sales orders to item fulfillments.

Sales Order Record Macro

The sales record type currently supports the autoAssignLocations macro. The Automatic Location Assignment engine processes the sales order and, based on some business rules, sets the best location on each line.

For help working with this record in the UI, see the help topic **Sales Orders**.

For information about SuiteScript 2.0 record macros, see the following help topics:

- **Overview of Record Action and Macro APIs**
- **N/record Module Members**
- **Macro Object Members**

<table>
<thead>
<tr>
<th>autoAssignLocations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Corresponding UI Button</strong></td>
</tr>
<tr>
<td><strong>Macro Description</strong></td>
</tr>
<tr>
<td><strong>Returns</strong></td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Since 2019.2

Macro Syntax

```javascript
require(‘/N/currentRecord’), function(currentRecord) {
    var salesOrder = currentRecord.get();

    salesOrder.executeMacro({id: ‘autoAssignLocations’});
};
```

Statistical Journal Entry

**Note:** This topic applies to all versions of SuiteScript.

The statistical journal entry record lets you increase or reduce the balance of a statistical account.

To use this record, the Statistical Accounts feature must be enabled at Setup > Enable Features, on the Accounting subtab. Also, you must have already created at least one statistical account. In the UI, you access this record at Transactions > Financial > Make Statistical Journal Entries.

For help working with this record in the UI, see the help topic Making Statistical Journal Entries.

This internal ID for this record is `statisticaljournalentry`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The statistical journal entry record is scriptable in both client and server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Supported Functions

The statistical journal entry record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

Be aware of the following:

- The unitlabel field is read-only. It is populated automatically when you set the unitstype field. (The unitlabel value is the unitstype's base unit value. This value is defined in the corresponding unit of measure record.)
The unitstype body field cannot be updated after the record is created.
The field labeled Amount in the UI is called debit in SuiteScript.
Every statistical journal entry record must have at least one line.
All sublist lines must use the same unit of measure, which is defined by the unitstype body field.

Code Samples

The following samples show how you can script to the statistical journal entry record.

Adding and Deleting

The following sample shows how you can create the record, add lines, load the record, and delete it.

```javascript
// Note: These samples use constants (defined like: var subsidiaryId = '1') which are not included here.

// Create Record
var statisticalJournal = record.create({
    type: record.Type.STATISTICAL_JOURNAL_ENTRY
});
statisticalJournal.setValue({
    fieldId: 'subsidiary',
    value: subsidiaryId
});
statisticalJournal.setValue({
    fieldId: 'externalid',
    value: externalId
});
statisticalJournal.setValue({
    fieldId: 'unitstype',
    value: unitsTypeId
});
statisticalJournal.setValue({
    fieldId: 'unit',
    value: unitId
});

// Add line to the record
statisticalJournal.setSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    line: 1,
    value: statisticalAccountId
});
statisticalJournal.setSublistValue({
    sublistId: 'line',
    fieldId: 'debit',          // field 'debit' has label 'Amount' in UI
    line: 1,
    value: amount
});
statisticalJournal.setSublistValue({
    sublistId: 'line',
    fieldId: 'lineunit',
    line: 1,
});
```
value: unitId
});
statisticalJournal.setSublistValue({
  sublistId: 'line',
  fieldId: 'memo',
  line: 1,
  value: memo
});
statisticalJournal.setSublistValue({
  sublistId: 'line',
  fieldId: 'class',
  line: 1,
  value: classId
});
statisticalJournal.setSublistValue({
  sublistId: 'line',
  fieldId: 'department',
  line: 1,
  value: departmentId
});
statisticalJournal.setSublistValue({
  sublistId: 'line',
  fieldId: 'location',
  line: 1,
  value: locationId
});

// Add record
var recId = statisticalJournal.save();

// Load record
var statisticalJournalAddedRec = record.load({
  type: record.Type.STATISTICAL_JOURNAL_ENTRY,
  id: recId
});

// Delete record
var deleteId = record.delete({
  type: record.Type.STATISTICAL_JOURNAL_ENTRY,
  id: recId
});

---

**Updating the Subsidiary Field**

The following sample shows how to update the subsidiary field, which and only be done in dynamic mode.

```javascript
var recType = record.Type.STATISTICAL_JOURNAL_ENTRY;

// 1st: Load record in Dynamic Mode
var journalEntry = record.load({
  type: recType,
  id: '168',
  isDynamic: true
});
```
// 2nd: Remove all lines
var lineCount = journalEntry.getLineCount({
    sublistId: 'line' });
for (i = 1; i <= lineCount; i++){
    var removeId = journalEntry.removeLine({
        sublistId: 'line',
        line: i });
}

// 3rd: Change subsidiary
journalEntry.setValue({
    fieldId: 'subsidiary',
    value: '6' });

// 4th: Add new line for changed subsidiary
// The new account must be available in the new subsidiary
journalEntry.selectNewLine({
    sublistId: 'line' });
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'account',
    value: '286' });
journalEntry.setCurrentSublistValue({
    sublistId: 'line',
    fieldId: 'debit',
    value: '5' });
journalEntry.commitLine({
    sublistId: 'line' });

var recId = journalEntry.save();

---

Store Pickup Fulfillment

Note: This topic applies to all versions of SuiteScript.

The store pickup fulfillment record is used when you fulfill an order from a location that allows store fulfillment and the fulfillment choice of line items in the order is Store Pickup.

For help working with this record in the UI, see the help topic Fulfilling Store Pickup Orders.

Note: If there are other line items in the sales order with the fulfillment choice set to Ship, you create an item fulfillment record for those lines. See Item Fulfillment for more information.

The internal ID for this record is storepickupfulfillment.
For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The store pickup fulfillment record is scriptable in both client and server SuiteScript.

**Supported Functions**

The store pickup fulfillment record supports the following functions: read, edit, transform, delete, and search.

**Note:** The create and copy functions are not supported.

**Usage Notes**

To use this record, you must have the Store Pickup feature enabled. See the help topic Store Pickup for more information about enabling and configuring the feature in NetSuite.

You should use the Store Pickup feature with the Fulfillment Request feature when possible. A fulfillment request is an intermediate record between the sales order and the store pickup fulfillment. With this intermediate record, you can better manage workload at a store location, as well as add fulfillment exceptions if there are problems fulfilling an item. With the Fulfillment Request feature enabled, you first create a sales order, then transform the sales order record to a fulfillment request record, and finally you transform the fulfillment request record to a store pickup fulfillment request record. See Example 1.

If the Fulfillment Request feature is not enabled, you transform the sales order record to a store pickup fulfillment record. See Example 2.

In both cases, the Fulfillment Choice field must be set to Store Pickup at the line-item level in the sales order.

**Code Samples**

**Example 1:** Create a store pickup fulfillment from a fulfillment request. The sales order ID is 321654. The fulfillment request ID is 221100 and is passed to the record.transform(options) API as a hash table.

```javascript
var fulfillmentRequestRecord = {"fftreqid" : 221100};
var storePickupFulfillRecord = record.transform({
    fromType: record.Type.SALES_ORDER,
    fromId: 321654,
    toType: record.Type.FULFILLMENT_REQUEST,
    defaultValues: fulfillmentRequestRecord
});
var storePickupFulfillRecordId = storePickupFulfillRecord.save();
```

**Example 2:** Create a store pickup fulfillment from a sales order.

```javascript
```
var storePickupFulfillRecord = record.transform({
  fromType: record.Type.SALES_ORDER,
  fromId: 321654,
  toType: record.Type.STORE_PICKUP_FULFILLMENT
});
var storePickupFulfillRecordId = storePickupFulfillRecord.save();

Subscription

Note: This topic applies to all versions of SuiteScript.

The subscription record is used to build recurring subscriptions.

The subscription record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features (Administrator), on the Transactions subtab. When the feature is enabled, you can access the subscription record in the UI by going to Transactions > Subscriptions > Create Subscriptions.

For help working with this record in the UI, see the help topic Subscription Management.

The internal ID for this record is subscription.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The subscription record is scriptable in server and client SuiteScript.

Supported Functions

The subscription record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Subscription Line

Note: This topic applies to all versions of SuiteScript.

The subscription line record is used to provide subscription item information for subscriptions.

The subscription line record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features (Administrator), on the Transactions subtab.
For help working with this record in the UI, see the help topic Subscription Management.

The internal ID for this record is subscriptionline.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The subscription line record is scriptable in server and client SuiteScript.

**Supported Functions**

The subscription line record is not fully scriptable — create and delete are not supported. Subscription lines can be created and deleted only from a subscription plan in the UI.

**Time**

**Note:** This topic applies to all versions of SuiteScript.

A time transaction, also known as TimeBill, records the hours worked by an employee. This transaction can be used to record billable hours and invoice customers. This transaction is available when the Time Tracking feature is enabled.

For help working with this record in the UI, see the help topic Managing Time Tracking.

The internal ID for this record is timebill.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Type**

The time record is scriptable in both client and server SuiteScript.
Supported Functions

The time record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

Time Actions

Note: This topic applies to all versions of SuiteScript.

The time record type, also known as timebill, currently supports the approve, reject, and submit actions.

For information about SuiteScript 2.0 record actions, see the following help topics:

- Overview of Record Action and Macro APIs
- N/action Module

approve

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Approve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action Description</td>
<td>Approve the timebill with the specified ID.</td>
</tr>
<tr>
<td>Returns</td>
<td>(&quot;id&quot;:recordId), &quot;notifications&quot;: [ Error, Warning, Information and text]</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>5 usage units</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Parameters

See the help topic action.execute(options) for details about parameters required for the execution of any action. The approve action does not support any additional parameters.

reject

<table>
<thead>
<tr>
<th>Corresponding UI Buttons</th>
<th>Reject</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Reject with note</td>
</tr>
<tr>
<td>Action Description</td>
<td>Reject the timebill with the specified ID. Optionally include a note that explains the reason for the rejection.</td>
</tr>
<tr>
<td></td>
<td>Available when the Advanced Approvals on Time preference is enabled.</td>
</tr>
<tr>
<td>Returns</td>
<td>(&quot;id&quot;:recordId), &quot;notifications&quot;: [ Error, Warning, Information and text]</td>
</tr>
</tbody>
</table>
Transactions

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>5 usage units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

### Parameters

See the help topic `action.execute(options)` for details about parameters required for the execution of any action. The reject action also supports the following additional parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>string</td>
<td>optional</td>
<td>An explanation of why the timebill was rejected.</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

### submit

<table>
<thead>
<tr>
<th>Corresponding UI Buttons</th>
<th>Submit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action Description</td>
<td>Submit the timebill with the specified ID for approval. Available when the Advanced Approvals on Time preference is enabled.</td>
</tr>
<tr>
<td>Returns</td>
<td>(“id”:recordId), “notifications”: [ Error, Warning, Information and text]</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>5 usage units</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

### Parameters

See the help topic `action.execute(options)` for details about parameters required for the execution of any action. The approve action does not support any additional parameters.

### Transaction Search

**Note:** This topic applies to all versions of SuiteScript.

Nearly all transaction record types use the TransactionSearch record for search.

For help working with this record in the UI, see the help topic Defining an Advanced Search. For help working with this record by using SuiteScript API, see the help topic N/search Module.

The internal ID for this record is `transaction`.

See the SuiteScript Records Browser for all internal IDs associated with this record.
**Transactions**

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/search Module
- N/record Module

**Supported Script Types**

The transaction search is scriptable in both client and server SuiteScript.

**Supported Functions**

Only search is supported for the transaction search record.

**Usage Notes**

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field</th>
<th>Internal ID</th>
<th>Field UI</th>
<th>Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>ccnumber</td>
<td></td>
<td>Credit Card #</td>
<td>Credit Card #</td>
<td>To prevent users from accessing sensitive information such as password and credit card data, this field cannot be read in beforeSubmit user event scripts for external role users (for example, shoppers, online form users (anonymous users), customer center).</td>
</tr>
<tr>
<td>entity</td>
<td></td>
<td>Name</td>
<td>Name</td>
<td>The search filter entity is synonymous for the search filter name. Either filter can be used when searching the value of the Name / ID field in the UI.</td>
</tr>
</tbody>
</table>

**Transfer Order**

**Note:** This topic applies to all versions of SuiteScript.

The Transfer Order transaction is used to move inventory between locations when the Multi-Location Inventory (MLI) feature is enabled. Existing integrations with external warehouse management systems can leverage this transaction to manage data about inventory movement between locations.

Transfer orders can initialize item fulfillment and item receipt transactions. See the help topics Item Fulfillment and Item Receipt for details about these transactions.

For help working with this record in the UI, see the help topics Order Fulfillment and Customer Return Management.

The internal ID for this record is `transferorder`. 
See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Usage Notes

### Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Scriptable Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search. When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
</tbody>
</table>

### Unlocked Time Period

**Note:** This topic applies to all versions of SuiteScript.

The unlocked time period record is available only when the Weekly Timesheets feature is enabled and the Lock Timesheet Period accounting preference is set to lock timesheets after a specified time.

For help working with timesheet locking in the UI, see the help topic Locking Timesheet Periods.

The internal ID for this record is unlockedtimeperiod.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Type

The unlocked time period record is scriptable in both client and server SuiteScript.
Supported Operations

This record can be created, read, updated, deleted, copied, and searched using SuiteScript. It cannot be transformed.

Usage Notes

When creating a new unlocked time period, you should not create a record for an employee that overlaps the date range of an existing active time period for the same employee.

The System Notes sublist is search only.

Field Definitions

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Code Samples

The following sample shows how to create an unlocked time period.

```javascript
require (['N/record', 'N/format'], function (record, format){
    var unlockTime = record.create ({
        type: record.Type.UNLOCK_TIME_PERIOD,
        isDynamic: true
    });
    unlockTime.setValue({
        fieldId: 'entity',
        value: -5
    });
    unlockTime.setValue({
        fieldId: 'startdate',
        value: format.parse({
            value: "6/7/2019",
            type: format.Type.DATE
        })
    });
    unlockTime.setValue({
        fieldId: 'enddate',
        value: format.parse({
            value: "6/9/2019",
            type: format.Type.DATE
        })
    });
    unlockTime.setValue({
        fieldId: 'validuntil',
        value: format.parse({
            value: "6/29/2019",
            type: format.Type.DATE
        })
    });
});
```
Usage

**Note:** This topic applies to all versions of SuiteScript.

The usage record is used to calculate usage information for subscriptions.

The usage record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features, on the Transactions subtab. When the feature is enabled, you can access the usage record in the UI by going to Transactions > Subscriptions > Create Usages.

For help working with this record in the UI, see the help topic Creating Usage Records.

The internal ID for this record is `usage`.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The usage record does not support client scripts, server scripts, or workflows.

**Supported Functions**

The usage record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Vendor Bill

**Note:** This topic applies to all versions of SuiteScript.

The vendor bill transaction records payables as they arrive from vendors, allowing you to pay bills from the payables list as they are due, and providing an accurate picture of payables at all points of the billing cycle.

For help working with this record in the UI, see the help topic Vendor Bills.

The internal ID for this record is `vendorbill`. 

SuiteScript Records Guide
See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The vendor bill record is scriptable in both client and server SuiteScript.

**Supported Functions**

The vendor bill record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When this field appears on the sublist line level, this field is not scriptable.</td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
<td>When this field is on the body of the form in edit mode, this field is scriptable and can be returned in a transaction search.</td>
</tr>
<tr>
<td>usertotal</td>
<td>Amount</td>
<td>This field is not available via search or lookup for any transactions.</td>
</tr>
</tbody>
</table>

**Using Landed Cost Fields**

When you create a landed cost category, the associated field IDs for the first category are landedcostamount1 and landedcostsource1. If you create a second category, the IDs will be landedcostamount2 and landedcostsource2.

This pattern increments by one with each additional category. For example, the IDs for the next landed cost category will be landedcostamount3 and landedcostsource3, and so on.
Using the Payee Address Sublist

As of 2019.2, payee address data is stored in a subrecord instead of a text field. On the Bill Payment form in the UI, address information is now displayed in a sublist instead of a single field. SuiteScript includes specialized APIs that you must use to script with subrecord data. You should use these subrecord APIs to script with payee address data.

For details about scripting subrecords with SuiteScript 2.0, see:

- SuiteScript 2.0 Scripting Subrecords
- Record Object Members
- CurrentRecord Object Members
- Understanding the Address Subrecord

**Warning:** This change may impact existing scripts that reference bill payee address data. You should review these scripts to determine whether updates are needed.

---

Vendor Credit

**Note:** This topic applies to all versions of SuiteScript.

A vendor credit transaction creates a credit, from a vendor, that can be applied to a payables account. For example, a vendor credit transaction may occur when items are returned to a vendor or when a discount is negotiated with a vendor.

For help working with this record in the UI, see the help topic Vendor Credits.

The internal ID for this record is `vendorcredit`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

---

Supported Script Types

The vendor credit record is scriptable in both client and server SuiteScript.

---

Supported Functions

The vendor credit record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.
Usage Notes

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>usertotal</td>
<td>Amount</td>
<td>This field is not available via search or lookup for any transactions.</td>
</tr>
</tbody>
</table>

Vendor Payment

**Note:** This topic applies to all versions of SuiteScript.

A vendor payment transaction posts to the general ledger as an expense and the amount of the payment is deducted from your accounts payable total. A vendor payment can be applied to one or more vendor bills. Vendor payments can help to track expenditures and total payables due.

For help working with this record in the UI, see Vendor Payments.

The internal ID for this record is vendorpayment.

See the SuiteScript Records Browser for all internal IDs associated with this record.

See the Vendor Payment Record Actions for the actions associated with this record. For more information about actions and macros, see the help topic Overview of Record Action and Macro APIs.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The vendor payment record is scriptable in both client and server SuiteScript.

Supported Functions

The vendor payment record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

The Credits sublist, scriptable for the vendor payment record, is a list sublist. In the NetSuite Help Center, see the help topic List Sublists for information on this sublist type.
Vendor Payment Record Actions

The vendor payment record currently supports the confirm and decline actions.

For help working with this record in the UI, see Vendor Payments.

For information about SuiteScript 2.0 record actions, see the following help topics:

- Overview of Record Action and Macro APIs
- N/action Module

**confirm**

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Confirm</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Description</strong></td>
<td>Changes a payment's status from Payment In-Transit to Payment Confirmed.</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td>(notifications[], response:{})</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client and server scripts</td>
</tr>
<tr>
<td></td>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>10 usage units</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Since</strong></td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Parameters**

See the help topic action.execute(options) for details about parameters required for the execution of any action. The confirm action supports the following parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>exchangerate</td>
<td>string</td>
<td>optional</td>
<td>The exchange rate at the time of the transaction's confirmation.</td>
<td>2018.2</td>
</tr>
<tr>
<td>confirmationdate</td>
<td>string</td>
<td>optional</td>
<td>The date on which the transaction is confirmed and posted.</td>
<td>2018.2</td>
</tr>
<tr>
<td>postingperiod</td>
<td>string</td>
<td>optional</td>
<td>The transaction posting period.</td>
<td>2018.2</td>
</tr>
<tr>
<td>clearpayment</td>
<td>string</td>
<td>optional</td>
<td>Indicates if the payment should be marked as cleared.</td>
<td>2018.2</td>
</tr>
<tr>
<td>recordId</td>
<td>int</td>
<td>required</td>
<td>The record ID.</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Vendor Payment Confirm Action Syntax

```javascript
require(['N/action'], function(action){
    // action & action list loading
    var actionList = action.find({
        recordType: 'vendorpayment'
    });
    var actionObj = action.get{
```
decline

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Decline</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action Description</strong></td>
<td>Declines a payment's status from Payment In-Transit to Payment Declined.</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td>(notifications[], response())</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client and server scripts</td>
</tr>
<tr>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>10 usage units</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Parameters**

See the help topic `action.execute(options)` for details about parameters required for the execution of any action. The `decline` action does not support parameters aside from `recordId`.
Vendor Payment Decline Action Syntax

```javascript
require(['N/action'], function(action){
    // action & action list loading
    var actionList = action.find({
        recordType: 'vendorpayment'
    });
    var actionObj = action.get({
        recordType: 'vendorpayment',
        id: 'decline'
    });

    // action execution 1
    var result = action.execute({
        recordType: 'vendorpayment',
        id: 'decline',
        params: {
            recordId: 101
        }
    });

    // action execution 2
    var result = actionObj.execute({
        recordId: 102
    });

    // action execution 3
    var result = actionObj({
        recordId: 103
    });
});
```

Vendor Prepayment

**Note:** This topic applies to all versions of SuiteScript.

A Vendor Prepayment is a posting transaction that impacts the general ledger without offsetting the Accounts Payable account. When the vendor prepayment is applied, the Accounts Payable account is offset. The Vendor Prepayments feature records and tracks deposit amounts paid to vendors before they accept a purchase order for a good or service. You can apply these prepayment amounts against open bills for the vendor.

In NetSuite the vendor prepayment can be associated to a purchase order or not associated to a purchase order (standalone vendor prepayments.)

For help working with this record in the UI, see the help topic *Entering Vendor Prepayments*.

The internal ID for this record is `vendorprepayment`.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
N/record Module

Supported Script Types

The vendor prepayment record is scriptable in both client and server SuiteScript.

Supported Fields

For all the internal IDs associated with this record, see the SuiteScript Records Browser.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Usage Notes

The script user must have the required permission level for the Vendor Prepayment permission. For information about vendor prepayment permissions, read the help topic Configuring Permissions for Vendor Prepayments.

Code Samples

The following example shows how to create a new vendor prepayment transaction for the payee with internal ID 94. The prepayment value is 500 units of the default payee currency:

### SuiteScript 1.0

```javascript
function createVendorPrepayment() {
    var entity = 94;
    var account = 151;
    var payment = 500;

    var vendorPrepayment = nlapiCreateRecord('vendorprepayment');
    vendorPrepayment.setFieldValues('entity', entity);
    vendorPrepayment.setFieldValues('account', account);
    vendorPrepayment.setFieldValues('payment', payment);
    id = nlapiSubmitRecord(vendorPrepayment);
}
```

### SuiteScript 2.0

```javascript
require(['N/record'], function(record) {
    function createAndSaveVendorPrepayment() {
        var objRecord = record.create({
            type: 'vendorprepayment',
            isDynamic: true
        });
        objRecord.setValue({
```
The following example shows how to update an existing vendor prepayment transaction with transaction ID 120:

**SuiteScript 1.0**

```javascript
function updateVendorPrepayment() {
    var id = 120;
    var memo = 'Updated transaction memo';
    var vendorPrepayment = nlapiLoadRecord('vendorprepayment', id);
    vendorPrepayment.setFieldValue('memo', memo);
    id = nlapiSubmitRecord(vendorPrepayment);
}
```

**SuiteScript 2.0**

```javascript
require(['N/record'], function(record) {
    function updateAndSaveVendorPrepayment() {
        var objRecord = record.load({
            type: 'vendorprepayment',
            id: 120,
            isDynamic: true,
        });

        objRecord.setValue({
            fieldId: 'memo',
            value: 'Updated transaction memo 2'
        });

        var recordId = objRecord.save();
    }
    updateAndSaveVendorPrepayment();
});
```

The following example shows how to delete a vendor prepayment transaction with transaction ID 120:

**SuiteScript 1.0**

```javascript
function deleteVendorPrepayment() {
    var id = 120;
    var vendorPrepayment = nlapiLoadRecord('vendorprepayment', id);
    vendorPrepayment.setFieldValue('delete', true);
    id = nlapiSubmitRecord(vendorPrepayment);
}
```
function deleteVendorPrepayment() {
  var id = 120;
  var vendorPrepayment = nlapiDeleteRecord('vendorprepayment', id);
}

---

**SuiteScript 2.0**

```javascript
require(['N/record'], function(record) {
  function deleteVendorPrepayment() {
    var objRecord = record.delete({
      type: 'vendorprepayment',
      id: 120
    });
    deleteVendorPrepayment();
  }
});
```

---

**Vendor Prepayment Application**

**Note:** This topic applies to all versions of SuiteScript.

After you make a prepayment to a vendor, you must apply the prepayment against the vendor bills. A vendor prepayment application is a posting transaction that impacts the general ledger. The Prepayment account is credited, offsetting the Accounts Payable account. The application decreases the total amount you must pay to your vendor.

You can apply the prepayment manually, or configure NetSuite to apply the prepayment automatically using the Auto-Apply accounting preference.

For help working with this record in the UI, see the help topic **Vendor Prepayment Application**.

The internal ID for this record is **vendorprepaymentapplication**.

For information about scripting with this record in SuiteScript, see the following help topics:

- Vendor Prepayment Application
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The vendor prepayment application record is scriptable in both client and server SuiteScript.

**Supported Fields**

For all the internal IDs associated with this record, see the SuiteScript Records Browser.
Usage Notes

The script user must have the required permission level for the Vendor Prepayment Application permission. For information about vendor prepayment permissions, read the help topic Configuring Permissions for Vendor Prepayments.

Code Samples

The following example shows how to transform a vendor prepayment transaction into the vendor prepayment application. The transformation process applies the vendor prepayment. This transformation is required because a vendor prepayment application requires a vendor prepayment.

- **SuiteScript 1.0**
  
  This example shows an application of the vendor prepayment with ID 120. During the application, the first vendor bill is selected for application.

  ```script
  function transformVendorPrepayment() {
      var id = 120;
      var vendorPrepaymentApplication = nlapiTransformRecord('vendorprepayment', id, 'vendorprepaymentapplication');
      vendorPrepaymentApplication.setLineItemValue('bill', 'apply', 1, 'T');
      var vendorPrepaymentApplicationId = nlapiSubmitRecord(vendorPrepaymentApplication);
  }
  ```

- **SuiteScript 2.0**
  
  This example shows an application of the vendor prepayment with ID 718. During the application, the first vendor bill is selected for application.

  ```script
  require(['N/record'], function(record) {
      function transformVendorPrepayment() {
          var vendorPrepaymentApplication = record.transform({
              fromType: 'vendorprepayment',
              fromId: 718,
              toType: 'vendorprepaymentapplication',
              isDynamic: true,
          });
          vendorPrepaymentApplication.selectLine({
              sublistId: 'bill',
              line: 0
          });
          vendorPrepaymentApplication.setCurrentSublistValue({
              sublistId: 'bill',
              fieldId: 'apply',
              value: true
          });
      }
  });
  ```

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.
A vendor return authorization is a non-posting transaction that tracks a return to a vendor, including the items to be returned, their quantities, the approval status, the shipment status, and the amount refunded or credited from the vendor. This type of transaction is available when the Vendor Return Authorizations feature is enabled.

The vendor return process includes four steps: creating a vendor return authorization record, approving or canceling the authorization, shipping items authorized to be returned, and crediting an authorized vendor return.

For help working with this record in the UI, see the help topic Vendor Return Authorization Overview. The internal ID for this record is vendorreturnauthorization. See the SuiteScript Records Browser for all internal IDs associated with this record.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The vendor return authorization record is scriptable in both client and server SuiteScript.

**Supported Functions**

The vendor return authorization record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

The following table provides usage notes for specific fields on this record.
<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>usertotal</td>
<td>Amount</td>
<td>This field is not available via search or lookup for any transactions.</td>
</tr>
</tbody>
</table>

**Weekly Timesheet**

*i Note: This topic applies to all versions of SuiteScript.*

The Weekly Timesheets feature works in conjunction with the existing Time Tracking feature to offer a customizable method of capturing time entries in a weekly format. Each timesheet represents a single week, and includes a sublist where each line represents a day of the week.

For help working with this record in the UI, see the help topic Weekly Timesheets.

The internal ID for this record is timesheet. In this newest version of the timesheet record, each sublist line representing a day of the week is a time record. The time record has an internal ID of timebill. In the deprecated version of the timesheet record, the values in the sublist are from a timeentry subrecord. The timeentry subrecord is not used by the newest version of the timesheet record.

See the SuiteScript Records Browser for all internal IDs associated with the timesheet record, and internal IDs associated with the Time record. At this time, the records browser entry for the timesheet record may include field IDs for the deprecated version of the record.

*i Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.*

To track time with weekly timesheets, an administrator should go to Setup > Enable Features > Employees, check the Time Tracking and Weekly Timesheets boxes, and click Save. To enable the optional new interface for Weekly Timesheets, go to Setup > Company > Setup Tasks > Enable Features (Administrator) > Employees. Check the New Weekly Timesheets Interface box, and click Save. After this feature is enabled, all users with timesheet permissions use the new UI. For more information, see Scripting Impact of New Weekly Timesheets Interface.

See Weekly Timesheet Record Actions and Weekly Timesheet Record Macros for the actions and macros associated with this record. For more information about actions and macros, see the help topic Overview of Record Action and Macro APIs.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

Both client and server side scripts are supported for the weekly timesheet record.

**Supported Functions**

The weekly timesheet record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.
Scripting Impact of New Weekly Timesheets Interface

NetSuite 2018.1 supports an optional new UI for weekly timesheets. For details, see the help topic Weekly Timesheets Interface.

If this new UI is enabled in your account, you may need to update your scripts on timesheets and time records, to ensure that they continue to work properly. The following changes in the new UI impact scripts on timesheet or time records:

- IDs for Popup Window Fields
- Changes to User Event Context Types
- Enforcement of Defined Time Limits

IDs for Popup Window Fields

The new UI allows some fields to be moved to a popup window for each day. Fields that are moved to the popup window have new field IDs for each day. By default, setting the value of a popup window field updates the field’s values for all days of the week, in other words, all timebills in a weekly timesheet. You can make it possible to update a field’s value for a single day by appending a number 0-6 to the end of the existing field ID. For example, memo0 represents the memo field for the first day of the week.

Client scripts on fields may need to be adjusted. Any scripts intended to change a field located in a popup window should be adjusted to accommodate new field IDs for single days of the week.

In addition, any scripts that are set to run on multiple forms that may or may not have the required fields in the expected locations are likely to fail. If you have client scripts that are set to run on multiple custom forms, you must ensure that the fields are located in the expected place on all forms. The preferred method is to use user event scripts instead of client scripts to update single timebills on a weekly timesheet.

Changes to User Event Context Types

When the new UI is not enabled, there are 3 separate user event execution contexts (context.UserEventType) for approving, rejecting, or submitting a timesheet. With the new UI enabled, all of these actions use the EDIT user event type, so you must update your user event scripts for timesheets and time records. Instead of checking for a specific execution context type, these scripts should check the Approval Status field to determine whether it has been changed.

Enforcement of Defined Time Limits

When the new UI is not enabled, time limits are only validated when time transactions are submitted through the UI. With the new UI enabled, time limits are validated when an individual time transaction is added using SuiteScript. You must ensure that time transactions submitted using SuiteScript meet defined time limits.

Code Samples

The following samples are for SuiteScript 2.0:

- Creating a New Weekly Timesheet
Creating a New Weekly Timesheet

```javascript
/**
 * @NapiVersion 2.x
 * @NScriptType ClientScript
 */
define(['N/record'], function(record) {
    var weeklyTimesheet = record.create({
        type: record.Type.TIME_SHEET,
        isDynamic: true
    });
    weeklyTimesheet.selectNewLine({
        sublistId: 'timeitem'
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'customer',
        value: '48'
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'item',
        value: '117'
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'location',
        value: '1'
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'hours0',
        value: '8'
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'memo',
        value: 'timeentry created'
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'isbillable',
        value: true
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'payrollitem',
        value: '2'
    });
    weeklyTimesheet.setCurrentSublistValue({
        sublistId: 'timeitem',
        fieldId: 'paidexternally',
        value: true
    });
});
```
weeklyTimesheet.setCurrentSublistValue({
    sublistId: 'timeitem',
    fieldId: 'price',
    value: '1'
});
weeklyTimesheet.setCurrentSublistValue({
    sublistId: 'timeitem',
    fieldId: 'overriderate',
    value: true
});
weeklyTimesheet.setCurrentSublistValue({
    sublistId: 'timeitem',
    fieldId: 'department',
    value: '1'
});
weeklyTimesheet.setCurrentSublistValue({
    sublistId: 'timeitem',
    fieldId: 'class',
    value: '1'
});
weeklyTimesheet.commitLine({
    sublistId: 'timeitem'
});

var weeklyTimesheetId = weeklyTimesheet.save();

Verifying Total Hours Upon Submit

```javascript
/**
 * @NapiVersion 2.x
 * @NScriptType ClientScript
 */
define(['N/ui/dialog', 'N/error'], function(dialog, error) {
    function saveRecord(context) {
        var timesheetRecord = context.currentRecord;

        if (timesheetRecord.getValue('totalhours') != '40') {
            dialog.alert({
                title: 'Alert',
                message: 'Total time is not equal to 40'
            });
            throw error.create({
                name: 'MISSING_REQ_ARG',
                message: 'Total time is not equal to 40'
            });
        }
        return true;
    }
    return {
        saveRecord: saveRecord
    };
});```
Updating a Custom Field on a Weekly Timesheet Based on Time Entry Field Values

```javascript
/**
 * @NapiVersion 2.x
 * @NScriptType ClientScript
 */
define(['N/record'], function(record) {
    /*
     * Populates custom field 'custrecord_am_timesheet_billable_hours' with sum of time entry 'hours'
     * values for time entries that are billable. Returns true to continue with record submit;
     */
    function timesheet_clientSaveRecord(context) {
        var timesheetRecord = context.currentRecord;
        var intSum = 0;
        var intNumberLines = timesheetRecord.getLineCount({
            sublistId: 'timeitem'
        });
        var arTimeGridColumns = ['hours0','hours1','hours2','hours3','hours4','hours5','hours6'];
        for (var intLineCounter = 0; intLineCounter < intNumberLines; intLineCounter++){
            for (var intDayCounter = 0; intDayCounter < arTimeGridColumns.length; intDayCounter++){
                var srcTimeEntry = timesheetRecord.getSublistValue({
                    sublistId: 'timeitem',
                    fieldId: arTimeGridColumns[intDayCounter],
                    line: intLineCounter
                });
                if (srcTimeEntry != ''){
                    var bBillable = timesheetRecord.getSublistValue({
                        sublistId: 'timeitem',
                        fieldId: 'isbillable',
                        line: intLineCounter
                    });
                    if (bBillable){
                        intSum += srcTimeEntry;
                    }
                }
            }
        }
        timesheetRecord.setValue({
            fieldId: 'custrecord_am_timesheet_billable_hours',
            value: intSum
        });
        return true;
    }
    return {
        saveRecord: timesheet_clientSaveRecord
    };
});
```
Weekly Timesheet Record Actions

The weekly timesheet record type currently supports the approve, reject, and submit actions. For help working with this record in the UI, see the help topic Weekly Timesheets. For information about SuiteScript 2.0 record actions, see the following help topics:

- Overview of Record Action and Macro APIs
- N/action Module

### approve

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Action Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approve All Pending</td>
<td>Approve all pending timebills that the user has permission to approve on the timesheet with the specified ID.</td>
<td>{&quot;id&quot;:recordId}, list containing all timeBills' &quot;notifications&quot;: [Error, Warning, Information and text]</td>
<td>Client and server scripts</td>
<td>5 usage units</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Parameters

See the help topic action.execute(options) for details about parameters required for the execution of any action. The approve action does not support any additional parameters.

### reject

<table>
<thead>
<tr>
<th>Corresponding UI Buttons</th>
<th>Action Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reject All Pending</td>
<td>Reject all pending timebills that the user has permission to reject on the timesheet with the specified ID. Optionally include a note that explains the reason for the rejection. Available when the Advanced Approvals on Time preference is enabled.</td>
<td>{&quot;id&quot;:recordId}, list containing all timeBills' &quot;notifications&quot;: [Error, Warning, Information and text]</td>
<td>Client and server scripts</td>
<td>5 usage units</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Parameters

See the help topic action.execute(options) for details about parameters required for the execution of any action. The reject action also supports the following additional parameter.
### Transactions

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>note</td>
<td>string</td>
<td>optional</td>
<td>An explanation of why the pending timebills were rejected.</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

#### submit

**Corresponding UI Buttons**

- Submit

**Action Description**

Submit open and rejected timebills on the timesheet with the specified ID. Available when the Advanced Approvals on Time preference is enabled.

**Returns**

```json
{"id":recordId}, list containing all timeBills' "notifications": [ Error, Warning, Information and text ]
```

**Supported Script Types**

Client and server scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

5 usage units

**Since**

2018.2

### Parameters

See the help topic `action.execute(options)` for details about parameters required for the execution of any action. The submit action does not support any additional parameters.

### Weekly Timesheet Record Macros

The weekly timesheet record currently supports the `checkTimeLimits` and `copyFromWeek` macros.

For help working with this record in the UI, see the help topic Weekly Timesheets.

For information about SuiteScript 2.0 record macros, see the following help topics:

- Overview of Record Action and Macro APIs
- N/record Module Members
- Macro Object Members

### checkTimeLimits

**Corresponding UI Button**

Check Time Limits

**Macro Description**

Gathers the contents of the timeitem machine and sends to a service. Based on preferences, returns the results of a server check of time limits: either indicates that limits were not exceeded or provides a list of errors.

**Returns**

Limit check notifications

See Sample Return Objects.

**Supported Script Types**

Client and server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.
Parameters

See the help topic `Record.executeMacro(options)` for details about parameters required for the execution of any macro. The checkTimeLimits macro also supports the following additional parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>string</td>
<td>optional</td>
<td>The action for which the limits are checked. May be one of the following: ■ approve ■ save ■ submit</td>
<td>2018.1</td>
</tr>
<tr>
<td>limitcheck</td>
<td>boolean</td>
<td>optional</td>
<td>Checks limits regardless of the action.</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Note: Based on preferences enabled in the account, limits may not be checked for the approve action. If the submit action is available limits are not checked for the save action.

Sample Return Objects

```javascript
// Limit check with some errors but allow the user to continue, with optional preferences setting
{
  notifications: [
    {
      title: "Maximum hours per day rule violated",
      message: "4/17/2017 has more than 10:00 hours",
      severity: {
        label = "Warning",
        value = 2
      }
    },
    {
      title: "Maximum hours per week rule violated",
      message: "Week has more than 40:00 hours",
      severity: {
        label = "Warning",
        value = 2
      }
    }
  ],
  rules: [
    {
      name: "MAX_PER_DAY",
      value: "8",
      action: "WARN"
    }
  ]
}
```
```json
{
    name: "MAX_PER_WEEK",
    value: "48",
    action: "DISALLOW"
}
```

// Limit check with no errors:
```json
{
    notifications: [
        {
            title: "Limit check successful",
            message: "All time records meet their respective rules",
            severity: {
                label: "Information",
                value: 1
            }
        }
    ]
}
```

Limit check that disallows continuation:
```json
{
    notifications: [
        {
            title: "Maximum hours per week rule violated",
            message: "Week has more than 40:00 hours",
            severity: {
                label: "Error",
                value: 3
            }
        }
    ]
}
```

**copyFromWeek**

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Copy From Week</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Macro Description</strong></td>
<td>Copies all lines from the specified week to the current weekly timesheet. Does not replace any lines in the weekly timesheet. Hours and memos may not be copied.</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td><code>{notifications=[], response:()}</code></td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client and server scripts For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Parameters**

See the help topic [Record.executeMacro(options)] for details about parameters required for the execution of any macro. The checkTimeLimits macro also requires the following additional parameters.
### Weekly Timesheet Macros Syntax

```javascript
require(['N/currentRecord', 'N/format'], function(currentRecord, format){
    var timesheet = currentRecord.get();

    // Copy time data from the week of 7/10, including hours and memos
    timesheet.executeMacro({
        id: 'copyFromWeek',
        params: {
            weekOf: '7/10/2017',
            copyExact: true
        }
    });

    // Copy time data from the week of 14/10, but do not include hours and memos
    timesheet.executeMacro({
        id: 'copyFromWeek',
        params: {
            weekOf: '14/10/2017',
            copyExact: false
        }
    });

    // Check whether the data currently in timesheet meet limits set by the administrator (for example, maximum of 40 hours per week, minimum of 6 hours per working day)
    var limitCheckResult = timesheet.executeMacro({
        id: 'checkTimeLimits',
        params: {
            action: 'submit'
        }
    });
});
```

### Work Order

**Note:** This topic applies to all versions of SuiteScript.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

Work order transactions track the production of assembly items needed for stock or to fill orders. Work orders track the quantities of assemblies that need to be built and the quantities of components, or
member items, needed to do so. This type of transaction is available when the Assembly Items and
Work Orders features are enabled and is used when the Allow Purchase of Assembly Items accounting
preference is not enabled.

Special order work orders track assemblies for a particular sale, and Production work orders track
assemblies to increase stock. Both use the same work order form, but Production work orders do not
link to a sales transaction. Production work orders are generated when the back ordered quantity of an
assembly reaches its assigned build point. After the build point is reached, a work order is added in the
Mass Create Work Orders queue.

For help working with this record in the UI, see the help topic Assembly Work Orders.

The internal ID for this record is workorder.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Additional Information:

For information on using the SuiteScript Records Browser, see the help topic Working with the
SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The work order record is scriptable in both client and server SuiteScript.

Supported Functions

The work order record is fully scriptable, which means that the record can be created, updated, copied,
deleted, and searched using SuiteScript.

Usage Notes

Fields

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>estgrossprofit</td>
<td>Est. Gross Profit</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>estgrossprofitpercent</td>
<td>Est. Gross Profit Percent</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>totalcostestimate</td>
<td>Est. Extended Cost</td>
</tr>
</tbody>
</table>

The Multi-Partner Management feature must be enabled in your account for the Partners sublist to
appear.
Work Order Close

Note: This topic applies to all versions of SuiteScript.

If the Manufacturing Work In Process (WIP) feature has been enabled, you can use SOAP web services to interact with work order close records. You can check to see if WIP is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

With WIP, instead of creating a single assembly build record to denote that a work order has been addressed, you track progress of the work using three records: work order issue, work order completion, and work order close. This approach lets you manage the assembly process in a more granular way, and to keep the General Ledger up to date as materials move through the different phases of assembly.

For details on the benefits of WIP, see the help topic Manufacturing Work In Process (WIP). For information about the process of closing a work order without WIP enabled, see the help topic Marking Work Orders Closed.

For help working with this record in the UI, see the help topic Entering Work Order Closes.

The internal ID for this record is `workorderclose`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The work order close record is scriptable in server SuiteScript only.

Supported Functions

The work order close record is partially scriptable. It can be updated, deleted, and searched using SuiteScript. It cannot be copied or created.

Usage Notes

To use this record you must have the following features enabled: Work Orders and Manufacturing Work in Process.

In the UI, this record is accessed by going to Transactions > Manufacturing > Close Work Order.

Note these additional details:

- You must use `record.transform(options)` to create a new instance of this record. In this case, `workorder` is the originating record type. For more details, see Prerequisites for Creating a Work Order Issue Record.
- Assembly item should have scrap account, WIP account, and WIP Cost Variance Account specified.
record.create(options) and record.copy(options) are not supported on this record.

**Prerequisites for Creating a Work Order Issue Record**

Before you can create a work order issue record, a work order record must already exist, and the work order must be configured to use WIP (the WIP box on the work order record must be selected). This is true regardless of whether you are creating the work order issue record using initialize and add, or add by itself. If you try to create a work order issue record referencing a work order that has not been configured to use WIP, the system generates an error reading in part, “One of the following problems exists: You have an invalid work order < work order ID >, the work order does not use WIP, or the work order is already closed.” You can create and modify work orders by choosing Transactions > Manufacturing > Enter Work Orders.

You can also interact with work orders using SuiteScript, as described in Work Order.

Note also that the assembly item referenced in the work order must be properly set up for WIP, as described in the Setting Up Items as WIP Assemblies.

**Work Order Completion**

![Note: This topic applies to all versions of SuiteScript.]

If the Manufacturing Work In Process (WIP) feature has been enabled, you can use SOAP web services to interact with work order completion records. You can check to see if WIP is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

With WIP, instead of creating a single assembly build record to denote that a work order has been addressed, you track progress of the work using three records: work order issue, work order completion, and work order close. This approach lets you manage the assembly process in a more granular way, and to keep the General Ledger up to date as materials move through the different phases of assembly. For more on the benefits of WIP, refer to Manufacturing Work In Process (WIP).

The work order completion record is used to indicate that assemblies have been built. You can also optionally use this record to record that raw materials — items in the componentList sublist — have been consumed as part of the assembly process. This latter option is called entering a completion with backflush. For example, you might enter a completion with backflush if previous records (such as work order issue) did not record the consumption of all the materials you ended up using.

In the UI, you can view the form used for creating the work order completion record by choosing Transactions > Manufacturing > Enter Completions, selecting a Subsidiary (for OneWorld accounts), then clicking Complete that corresponds with one of the listed work orders. An alternate method is to view the work order and click one of two buttons: Enter Completions or Enter Completions With Backflush.

For help working with this record in the UI, see the help topic Entering Work Order Completions.

The internal ID for this record is workordercompletion.

See the SuiteScript Records Browser for all internal IDs associated with this record.

![Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.]

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
Supported Script Types

The work order completion record is scriptable in server SuiteScript only.

Supported Functions

The work order completion record is partially scriptable. It can be updated, deleted, and searched using SuiteScript. It cannot be copied or created.

Usage Notes

To use this record you must have the following features enabled: Work Orders and Manufacturing Work in Process.

In the UI, this record is accessed by going to Transactions > Manufacturing > Enter Completions.

Note these additional details:

- You must use `record.transform(options)` to create a new instance of this record. In this case, `workorder` is the originating record type. For more details, see Prerequisites for Creating a Work Order Issue Record.
- Assembly item should have scrap account and WIP account specified.
- `record.create(options)` and `record.copy(options)` are not supported on this record.
- The Component sublist is available only when backflush = true

Prerequisites for Creating a Work Order Issue Record

Before you can create a work order issue record, a work order record must already exist, and the work order must be configured to use WIP (the WIP box on the work order record must be selected). This is true regardless of whether you are creating the work order issue record using initialize and add, or add by itself. If you try to create a work order issue record referencing a work order that has not been configured to use WIP, the system generates an error reading in part, “One of the following problems exists: You have an invalid work order < work order ID >, the work order does not use WIP, or the work order is already closed.” You can create and modify work orders by choosing Transactions > Manufacturing > Enter Work Orders.

You can also interact with work orders using SuiteScript, as described in Work Order.

Note also that the assembly item referenced in the work order must be properly set up for WIP, as described in the Setting Up Items as WIP Assemblies.

Work Order Issue

Note: This topic applies to all versions of SuiteScript.

If the Manufacturing Work In Process (WIP) feature has been enabled, you can use SOAP web services to interact with work order completion records. You can check to see if WIP is enabled by going to Setup > Company > Enable Features, and reviewing the Items & Inventory tab.

With WIP, instead of creating a single assembly build record to denote that a work order has been addressed, you track progress of the work using three records: work order issue, work order completion,
and work order close. This approach lets you manage the assembly process in a more granular way, and to keep the General Ledger up to date as materials move through the different phases of assembly. For more on the benefits of WIP, refer to Manufacturing Work In Process (WIP).

The work order issue record is used to indicate that particular quantities of raw materials, or component items, have been gathered for the production of the assembly item (or items).

In the UI, you can view the form used for creating the work order issue record by choosing Transactions > Manufacturing > Issue Components, selecting a Subsidiary (for OneWorld accounts), then clicking Issue that corresponds with one of listed work orders. An alternate method is to view the work order and click the Issue Components button.

For help working with this record in the UI, see the help topic Entering Work Order Issues.

The internal ID for this record is workorderissue.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The work order issue record is scriptable in server SuiteScript only.

**Supported Functions**

The work order issue record is partially scriptable. It can be updated, deleted, and searched using SuiteScript. It cannot be copied or created.

**Usage Notes**

To use this record you must have the following features enabled: Work Orders and Manufacturing Work in Process.

In the UI, this record is accessed by going to Transaction > Manufacturing > Issue Components.

Note these additional details:

- You must use record.transform(options) to create a new instance of this record. In this case, workorder is the originating record type. For more details, see Prerequisites for Creating a Work Order Issue Record.
- record.create(options) and record.copy(options) are not supported on this record.

**Prerequisites for Creating a Work Order Issue Record**

Before you can create a work order issue record, a work order record must already exist, and the work order must be configured to use WIP (the WIP box on the work order record must be selected).
This is true regardless of whether you are creating the work order issue record using initialize and add, or add by itself. If you try to create a work order issue record referencing a work order that has not been configured to use WIP, the system generates an error reading in part, “One of the following problems exists: You have an invalid work order < work order ID >, the work order does not use WIP, or the work order is already closed.” You can create and modify work orders by choosing Transactions > Manufacturing > Enter Work Orders.

You can also interact with work orders using SuiteScript, as described in Work Order.

Note also that the assembly item referenced in the work order must be properly set up for WIP, as described in Setting Up Items as WIP Assemblies.

Support

Note: This topic applies to all versions of SuiteScript.

The following records are scriptable in SuiteScript:

- Case
- Issue
- Issue Product
- Issue Product Version
- Solution
- Topic

Case

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Creating Case Types.

The internal ID for this record is supportcase.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The case record is scriptable in both server and client SuiteScript.
Supported Functions

The case record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Issue

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Logging Issues.

The internal ID for this record is `issue`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The issue record is scriptable in both server and client SuiteScript.

Supported Functions

The issue record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Issue Product

**Note:** This topic applies to all versions of SuiteScript.

The use of products on issue records helps determine which product team is responsible for the resolution of that issue.

For help working with this record in the UI, see the help topic Products and Modules.

The internal ID for this record is `issueproduct`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Supported Script Types

The issue record is scriptable in both server and client SuiteScript.
Supported Functions

The record can be created, read, updated and deleted using SuiteScript. It cannot be copied, transformed or searched.

Code Sample

The following script example illustrates how to create an issue product record.

```javascript
var product = record.create({
    type: record.Type.ISSUE_PRODUCT
});
product.setFieldValue({
    fieldId: 'issueproduct',
    value: 'new product'
});
var productId = product.save();
```

Issue Product Version

**Note:** This topic applies to all versions of SuiteScript.

If you track multiple versions of the same product, you can create versions and builds using the product version record.

For help working with this record in the UI, see the help topic Product Versioning.

The internal ID for this record is `productversion`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Supported Script Types

The issue record is scriptable in both server and client SuiteScript.

Supported Functions

The record can be created, read, updated and deleted using SuiteScript. It cannot be copied, transformed or searched.

Code Sample

The following script example illustrates how to create an issue product version record for a previously created issue product.

```javascript
var version = record.create({
    type: record.Type.ISSUE_PRODUCT_VERSION
});
version.setFieldValue({
    fieldId: 'issueproduct',
    value: productId
});
version.setFieldValue({
    fieldId: 'productVersion',
    value: versionId
});
```
The following script example illustrates how to add IssueProductBuild to IssueProductVersion.

```javascript
var version = record.load({
    type: record.Type.ISSUE_PRODUCT_VERSION,
    id: versionId // defined elsewhere
});
version.selectNewLine({
    sublistId: 'availablebuild'
});
version.setCurrentSublistValue({
    sublistId: 'availablebuild',
    fieldId: 'build',
    value: 'new build'
});
version.setCurrentSublistValue({
    sublistId: 'availablebuild',
    fieldId: 'description',
    value: 'Build description'
});
... version.commitLine({
    sublistId: 'availablebuild'
});
var recordId = version.save();
```

Solution

**Note:** This topic applies to all versions of SuiteScript.

The internal ID for this record is **solution**.

This record is available when the Knowledge Base feature is enabled at Setup > Company > Enable Features, on the CRM tab. When the feature is enabled, you can access the inventory cost revaluation record in the UI by choosing Lists > Support > Solutions > New.

For help working with this record in the UI, see the help topic Creating Knowledge Base Solutions.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
Supported Script Types

The solution record is scriptable in both server and client SuiteScript.

Supported Functions

The solution record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Topic

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Creating Knowledge Base Topics.

The internal ID for this record is `topic`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The topic record is scriptable only in server SuiteScript.

Supported Functions

The topic record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

File Cabinet

**Note:** This topic applies to all versions of SuiteScript.

The following records are scriptable in SuiteScript:
File

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the user interface, see the help topic Working with the File Cabinet.

The internal ID for this record is `file`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The file cabinet record is scriptable in server SuiteScript only.

Supported Functions

This record is not fully scriptable. Only search is permitted.

Usage Notes

Client SuiteScript is not supported for this record. It is scriptable in server SuiteScript only.

The file record is different from other NetSuite records. It has its own SuiteScript API. For information about the API for files, see the help topic N/file Module.

Folder

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Working with File Cabinet Folders.

The internal ID for this record is `folder`.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The folder record is scriptable in server SuiteScript only.

Supported Functions

The folder record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Lists

Note: This topic applies to all versions of SuiteScript.

The following list records are scriptable in SuiteScript:

- Account
- Accounting Book
- Accounting Context
- Accounting Period
- Allocation Schedule
- Amortization Schedule
- Amortization Template
- Billing Account
- Billing Class
- Billing Rate Card
- Billing Schedule
- Bin
- BOM
- BOM Revision
- Budget Exchange Rate
- Class
- Consolidated Exchange Rate
- Contact Category
- Contact Role
- Cost Category
- Currency
- Customer Category
- Customer Message
- Customer-Subsidiary Relationship
- Department
- Employee Change Request
- Employee Change Request Type
- Fair Value Price
- Financial Institution
- Format Profile
- Gift Certificate
- Global Account Mapping
- Global Inventory Relationship
- Government-Issued ID Tracking
  - Driver's License
  - Passport
  - Other Government-Issued ID
- Group
- Inbound Shipment
- Intercompany Allocation Schedule
- Inventory Number
- Item Account Mapping
- Item Collection
- Item Collection Item Map
- Item Location Configuration
- Item Revision
- Kudos Feature Records
  - Kudos Feature Records
  - Organization Value
- Location
- Manufacturing Cost Template
- Manufacturing Routing
- Merchandise Hierarchy Level
- Merchandise Hierarchy Node
- Merchandise Hierarchy Version
- Nexus
- Note Type
- Other Name Category
Account

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Chart of Accounts Management.
The internal ID for this record is **account**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

### Supported Script Types

The accounting book record is supported in server SuiteScript only.

### Supported Functions

The account record is fully scriptable using server SuiteScript.

#### Accounting Book

**Note:** This topic applies to all versions of SuiteScript.

When the Multi-Book Accounting feature has been enabled, you can use the accounting book record to create secondary books.

For help working with this record in the UI, see the help topic [Using Multi-Book Accounting](#).

In the UI, you access the accounting book record at **Setup > Accounting > Accounting Books > New**.

The internal ID for this record is **accountingbook**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

### Supported Script Types

The accounting book record is supported in server SuiteScript only.
All three user events are supported: **beforeLoad**, **beforeSubmit**, and **afterSubmit**.

### Supported Functions

This record is partially scriptable — it can be created, updated, deleted, and searched using SuiteScript.

### Code Samples

The following samples show how to create accounting book records and perform other basic tasks.

```javascript
// Create Record
var myACBook = record.create({
  type: record.Type.ACCOUNTING_BOOK
});
myACBook.setValue({
  fieldId: 'name',
  value: 'test'
});

var subs = new Array();
subs[0] = '1';
subs[1] = '3';
subs[2] = '5';

myACBook.setValue({
  fieldId: 'subsidiary',
  value: subs
});
myACBook.setValue({
  fieldId: 'isprimary',
  value: true
});

var recCreated = myACBook.save();

// Update Record
var myACBook = record.load({
  type: record.Type.ACCOUNTING_BOOK,
  id: '3'
});

var subs = new Array();
subs[0] = '1';
subs[1] = '3';
subs[2] = '5';

myACBook.setValue({
  fieldId: 'subsidiary',
  value: subs
});
myACBook.setValue(
```
Accounting Context

Note: This topic applies to all versions of SuiteScript.

An accounting context can be a one-to-one relationship between a country's local GAAP (Generally Accepted Accounting Principles) reporting requirements and a statutory chart of accounts (COA). It can also be a unique relationship that meets your company's specific needs. Accounting contexts are useful when users prefer to work in a local GAAP context, rather than in the consolidated context with one centralized COA. Accounting contexts are also useful if you have Multi-Book Accounting provisioned in your account. You can set an accounting context specific for your secondary book and use it for your secondary book reports.

For help working with this record in the UI, see the help topic Accounting Contexts.

The internal ID for this record is accountingcontext.

The accounting context record is available in OneWorld accounts.

The script user must have the Setup Company permission with Full permission level.

In the UI, you configure accounting contexts on the Accounting Contexts subtab on the General Preferences page at Setup > Company > General Preferences. If you do not have a OneWorld account, this subtab is not visible.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

This record is scriptable in both client and server SuiteScript.
Supported Operations

This record is partially scriptable — it can be read, created, deleted, copied, searched, and edited. It cannot be transformed.

Code Samples

The following samples show how to create an accounting context.

```javascript
require(['N/record'], function(record){
  var rec = record.create({
    type: record.Type.ACCOUNTING_CONTEXT
  });
  rec.setValue({
    fieldId: 'name',
    value: 'Context 1'
  });
  var recId = rec.save();
});
```

Accounting Period

Note: This topic applies to all versions of SuiteScript.

When the Accounting Periods feature has been enabled, this record is available to SuiteScript.

In the UI, you access accounting periods at Setup > Accounting > Manage Accounting Periods.

For help working with this record in the UI, see the help topic Accounting Period Setup.

The internal ID for this record is `accountingperiod`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The accounting period record is supported in client and server SuiteScript, for all script types.

Supported Functions

Only search and read functions are supported.
Code Sample

```javascript
var period = record.load({
    type: record.Type.ACCOUNTING_PERIOD,
    id: '1'
});
var startDate = period.getValue({
    fieldId: 'startdate'
});
var endDate = period.getValue({
    fieldId: 'enddate'
});
```

Allocation Schedule

**Note:** This topic applies to all versions of SuiteScript.

Allocation schedules transfer balances from expense accounts into one or more other accounts.

The internal ID for this record is `allocationschedule`.

The allocation schedule record is available only when the Accounting Periods and Expense Allocation features are enabled. An administrator can enable these features at Setup > Company > EnableFeatures > Accounting subtab. For information about expense allocation, see the help topic Expense Allocation Overview.

For help working with this record in the UI, see the help topic Creating Expense Allocation Schedules.

If the Statistical Accounting feature is enabled, the allocation schedule record includes fields that enable you to base the weight for the allocation on the balance of a statistical account through statistical journals or as an absolute value. If the Dynamic Allocation feature is also enabled, the weight is dynamically calculated when the allocation journal is generated. An administrator can enable these features at Setup > Company > EnableFeatures > Accounting under Advanced Features. For information about statistical allocation schedules, see the help topic Working with Allocation Schedules Weighted by the Balance of a Statistical Account.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is scriptable in both client and server SuiteScript.

If the Dynamic Allocation feature is enabled, the currency field is not scriptable. None of the fields from the History subtab on the allocation schedule record in the UI are not scriptable.
**Supported Operations**

This record is partially scriptable — it can be read, created, deleted, copied, and updated. It cannot be transformed.

**Code Samples**

The following samples show how to create a statistical allocation schedule that divides the balance of one account and transfers the amounts into two other accounts.

```javascript
var myAllocSched = record.create(
    type: record.Type.ALLOCATION_SCHEDULE,
    defaultValues: false
);

myAllocSched.setValue(
    fieldId: 'name',
    value: 'Allocation Schedule A'
);

myAllocSched.setValue(
    fieldId: 'subsidiary',
    value: '1'
);

myAllocSched.setSublistValue(
    sublistId: 'allocationsource',
    fieldId: 'account',
    line: 1,
    value: '24'
);

myAllocSched.setSublistValue(
    sublistId: 'allocationdestination',
    fieldId: 'account',
    line: 1,
    value: '6'
);

myAllocSched.setSublistValue(
    sublistId: 'allocationdestination',
    fieldId: 'weight',
    line: 1,
    value: '50.00'
);

myAllocSched.setSublistValue(
    sublistId: 'allocationdestination',
    fieldId: 'account',
    line: 1,
    value: '32'
);

myAllocSched.setSublistValue(
    sublistId: 'allocationdestination',
    fieldId: 'weight',
    line: 1,
    value: '50.00'
);
```
recordId = myAllocSched.save();

Amortization Schedule

Note: This topic applies to all versions of SuiteScript.

Amortization schedules are automatically created by the system for transactions that contain item or expense lines associated with amortization templates.

The internal ID for this record is amortizationschedule.

The amortization schedule record is available only when the Amortization feature is enabled, at Setup > Enable Features, on the Accounting subtab. In the UI, you access this record at Lists > Accounting > Amortization Schedules.

For help working with this record in the UI, see the help topic Amortization Schedules.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

This record is scriptable in server SuiteScript only.

The user events are not supported.

Supported Functions

This record is partially scriptable — it can be read, searched, and edited. It cannot be copied, created, deleted, or transformed.

Note: Edits are allowed only when the Allow Users to Modify Amortization Schedules preference has been selected. For details, see the help topic Setting Amortization Preferences.

Usage Notes

Be aware of the following:
- As in the UI, you cannot use external ID as part of your criteria when searching for amortization schedule records. You also cannot include external ID in search columns.
The amortization schedule record is similar to the revenue recognition schedule record — so if you have existing integrations for revenue recognition schedules, you may be able to reuse elements of these scripts. The revenue recognition schedule record is described in Revenue Recognition Schedule.

Amortization Template

**Note:** This topic applies to all versions of SuiteScript.

You use the amortization template record to define the terms of amortization schedules, which are automatically created by the system. For example, on certain transactions, you can associate an amortization template with a line in the item or expense sublist. When the transaction is saved, an amortization schedule based on the template is automatically created. You can use amortization templates in conjunction with journal entry, vendor bill, and vendor credit records.

The internal ID for this record is `amortizationtemplate`.

The amortization template record is available only when the Amortization feature is enabled, at Setup > Enable Features, on the Accounting subtab. In the UI, you access this record at Lists > Accounting > Amortization Templates > New.

For help working with amortization templates in the UI, see the help topic Creating Amortization Templates.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The amortization template record is supported in server SuiteScript only.

The user events are not supported.

**Supported Operations**

This record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

Be aware of the following:

- If the Method field (recurrencetype) is set to Custom, then you must include at least one line in the recurrence sublist. In this case, the template cannot be deleted.
- As in the UI, you cannot use external ID as part of your criteria when searching for amortization template records. You also cannot include external ID in search columns.
The amortization template record is similar to the revenue recognition template record — so if you have existing integrations for revenue recognition templates, you may be able to reuse elements of these scripts. The revenue recognition template record is described in Revenue Recognition Template.

## Code Sample

The following sample shows how to create amortization templates and perform other basic tasks.

```javascript
// Create Record
var recAmTemp = record.create({
    type: record.Type.AMORTIZATION_TEMPLATE
});
recAmTemp.setValue({
    fieldId: 'name',
    value: 'Name'
});
recAmTemp.setValue({
    fieldId: 'externalid',
    value: 'externalId'
});
recAmTemp.setValue({
    fieldId: 'isamortization',
    value: true
});
recAmTemp.setValue({
    fieldId: 'amortizationtype', // Type
    value: 'STANDARD'
});
recAmTemp.setValue({
    fieldId: 'recurrencetype', // Method
    value: 'EVENPERIODSPRORATE'
});
recAmTemp.setValue({
    fieldId: 'recognitvalsorc', // Term Source
    value: 'RECEIPTDATE'
});
recAmTemp.setValue({
    fieldId: 'revrecoffset',
    value: 0
});
recAmTemp.setValue({
    fieldId: 'periodoffset',
    value: 1
});
recAmTemp.setValue({
    fieldId: 'acctdeferral',
    value: '1'
});
recAmTemp.setValue({
    fieldId: 'acctcontra',
    value: '1'
});
recAmTemp.setValue({
    fieldId: 'accttarget',
    value: '1'
});
```
Billing Account

**Note:** This topic applies to all versions of SuiteScript.

The internal ID for this record is **billingaccount**. Full support is provided for client and server scripting with this record.

For help working with this record in the UI, see the help topic Creating Billing Accounts.

See the [SuiteScript Records Browser](https://help套件.net) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](https://help套件.net)
Supported Script Types

The billing account record is scriptable in both client and server SuiteScript.

Supported Functions

The billing account record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Billing Class

**Note:** This topic applies to all versions of SuiteScript.

You can use the billing class record to create different rates that you can use when calculating the cost of a resource's time on a project. You can use SuiteScript to create, update, and delete billing class records.

The internal ID for this record is `billingclass`.

The billing class record is available when the Per-Employee Billing Rates feature is enabled at Setup > Company > Enable Features, on the Employees tab. When the feature is enabled, you can access the billing class record in the UI by choosing Setup > Accounting > Billing Classes > New.

For help working with this record in the UI, see the help topic Using Billing Classes.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The billing class record is scriptable in server SuiteScript only.

All three user events are supported: `beforeLoad`, `beforeSubmit`, and `afterSubmit`.

Supported Functions

The billing class record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Field Definitions

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.
**Code Sample**

The following sample shows how to create a billing class record.

```javascript
var billingClass = record.create({
    type: record.Type.BILLING_CLASS
});

billingClass.setValue({
    fieldId: 'name',
    value: 'Billing Class'
});

billingClass.setValue({
    fieldId: 'description',
    value: 'Billing Class Description'
});

billingClass.setValue({
    fieldId: 'isinactive',
    value: false
});

billingClass.setSublistValue({
    sublistId: 'pricecost',
    fieldId: 'price',
    line: 1,
    value: 2.56
});
```

The following sample shows how to delete a billing class record.

```javascript
record.delete({
    type: record.Type.BILLING_CLASS,
    id: recId
});
```

**Billing Rate Card**

- **Note:** This topic applies to all versions of SuiteScript.

The internal ID for this record is `billingratecard`.

If you use billing classes, you can also use billing rate cards to define different billing rates for groups of billing classes. These rate cards can then be used to set billing rates on charge-based projects using time-based charge rules. You can use SuiteScript to read, create, update, delete, and search billing rate card records.

The billing rate card record is available when the Billing Rate Cards feature is enabled at Setup > Company > Enable Features, on the Employees tab. The Per-Employee Billing Rates and Charge-Based Billing features are also required to use billing rate cards. When the feature is enabled, you can access the billing rate card record in the UI by choosing Setup > Accounting > Billing Rate Cards > New.

For help working with this record in the UI, see the help topic Using Billing Rate Cards.

See the SuiteScript Records Browser for all internal IDs associated with this record.
See Billing Rate Card Record Macros for the macros associated with this record. For more information about actions and macros, see the help topic Overview of Record Action and Macro APIs.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The billing account record is scriptable in both client and server SuiteScript.

Supported Functions

The following SuiteScript actions are supported for this record:

- Read
- Create
- Edit
- Delete
- Search

Usage Notes

There are two sublists for rate card pricing: ratecardpricing and ratecardpricingmulti. These two lists are mutually exclusive. The first list, ratecardpricing, is a simple sublist used when the Multiple Currencies feature is not enabled. The second list, ratecardpricingmulti, is a matrix sublist used when the Multiple Currencies feature is enabled.

When this feature is enabled, the content of ratecardpricing is a subset of ratecardpricingmulti. If you enter information into ratecardpricing and then enable the Multiple Currencies feature, the original content is in a column in the ratecardpricingmulti sublist.

Note: Copy and transform are not supported.

Code Samples

The following sample shows how to create a new billing rate card with a single billing class without multiple currencies.

```javascript
var myBillingRate = record.create({
  type: record.Type.BILLING_RATE_CARD
});
myBillingRate.setValue({
  fieldId: 'name',
  value: 'abcd'
});
```
The following sample shows how to update the billing rate card name and a price in British pounds when multiple currencies is enabled.

```javascript
var myBillingRate = record.load({
  type: record.Type.BILLING_RATE_CARD,
  id: 3
});
var name = myBillingRate.getValue({
  fieldId: 'name'
});
var updatedName = name + ' - updated';
myBillingRate.setValue({
  fieldId: 'name',
  value: updatedName
});
myBillingRate.setSublistValue({
  sublistId: 'ratecardpricingmulti',
  fieldId: 'price_2_',
  line: 1,
  value: 5
});
var recId = myBillingRate.save();
```

The following sample shows how to use a user event script to update a price.

```javascript
// To get a new record in SuiteScript 2.0, use the following code in a beforeLoad, beforeSubmit, // or afterSubmit user event script:
function afterSubmit(context) {
  var newRec = context.newRecord;
}
// then use the new record as follows:
var origPrice = newRec.getSublistValue({
  sublist: 'ratecardpricing',
  fieldid: 'price',
  line: 1
});

var newPrice = 11;
newRec.setSublitValue({
  sublistId: 'ratecardpricing',
  fieldId: 'price',
  line: 1,
  value: 'newPrice'
});
log.debug({
```
Billing Rate Card Record Macros

Note: This topic applies to all versions of SuiteScript.

The billing rate card record type currently supports the `modifyPriceByPercent` macro.

For help working with this record in the UI, see the help topic [Using Billing Rate Cards](#).

For information about SuiteScript 2.0 record macros, see the following help topics:

- [Overview of Record Action and Macro APIs](#)
- [N/record Module Members](#)
- [Macro Object Members](#)

**modifyPriceByPercent**

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Recalculate</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Macro Description</strong></td>
<td>Modifies the price list on the record by a specified percentage.</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td><code>{notifications:[], response:{}}</code></td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
<td></td>
</tr>
</tbody>
</table>

**Since** 2018.1

**Parameters**

See the help topic [Record.executeMacro(options)](#) for details about parameters required for the execution of any macro. The `modifyPriceByPercent` macro also requires the following additional parameter.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>percent</td>
<td>number</td>
<td>required</td>
<td>The percentage amount by which the price list is modified. May be positive or negative.</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

**Billing Rate Card modifyPriceByPercent Macro Syntax**

The following code sample is from a Suitelet that increases the prices on a billing rate card with id 1 by 10%.

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptName Script Parameter Test
 * @NScriptType suitelet
 */
define(['N/record'], function(record) {
    return Object.freeze(
```
Billing Schedule

<i>Note:</i> This topic applies to all versions of SuiteScript.

This record enables you to create a billing schedule that can be applied to a sales order, a line item on a sales order, or a project. SuiteScript supports all five types of billing schedules (charge-based, fixed bid interval, fixed bid milestone, standard, and time and materials).

The internal ID for this record is <strong>billingschedule</strong>.

To use the billing schedule record, you must enable the Advanced Billing feature, at Setup > Enable Features, on the Transactions subtab. In the UI, you access this record at Lists > Accounting > Billing Schedules > New. You can access a billing schedule of fixed bid milestone type through the project record's Financial tab.

For help working with this record in the UI, see the help topic <a>Billing Schedules</a>.

See the <a>SuiteScript Records Browser</a> for all internal IDs associated with this record.

<i>Note:</i> For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

Billing Schedule is scriptable in server SuiteScript only.

All three user events are supported: <i>beforeLoad</i>, <i>beforeSubmit</i>, and <i>afterSubmit</i>.

**Supported Functions**

This record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.
Usage Notes

Additionally, note the following:

- To set a value for schedule type, which is a required body field, you must use initializeValues, not setValue. For examples, see the following section.
- If you choose a schedule type of fixed bid milestone, you must identify an existing project record (or job record). You do so using initializeValues. Then, to create a link between the project and the new billing schedule, you must update the project record — this relationship is not established automatically when you create the billing schedule.
- The Recurrence sublist is available only when schedule type is set to Standard and frequency to Custom.
- The Milestone sublist is available only when schedule type is set to Fixed Bid Milestone.

Code Samples

The following samples show how to create different types of billing schedules.

Creating a Charge-Based Billing Schedule

The following sample shows how to create a charge-based billing schedule.

```javascript
var SCHEDULE_TYPE = 'CB'; // Charge-Based
var recName = SCHEDULE_TYPE + 'record';

var initValues = new Array();
    initValues.schedtype = SCHEDULE_TYPE;

// Create record
var myBillingSched = record.create({
    type: record.Type.BILLING_SCHEDULE,
    defaultValues: {
        'schedtype': SCHEDULE_TYPE
    }
});
myBillingSched.setValue({
    fieldId: 'externalid',
    value: 'EXTID001'
});
myBillingSched.setValue({
    fieldId: 'name',
    value: recName
});
myBillingSched.setValue({
    fieldId: 'frequency',
    value: 'DAILY'
});
myBillingSched.setValue({
    fieldId: 'dayperiod',
    value: '3'
});
```
Creating a Fixed Bid Milestone Billing Schedule

The following sample shows how to create a billing schedule of fixed bid milestone type. Note that this sample references a particular project record during the creation of the billing schedule. However, you still have to establish the relationship between the billing schedule and the project as a separate step.

```javascript
var SCHEDULE_TYPE = 'FBM'; // Fixed Bid, Milestone
var P1 = '117';     // Project1
var P1M1 = '112';   // Project1 - Milestone1
var P1M2 = '113';   // Project2 - Milestone2
var recId = null;
var recName = SCHEDULE_TYPE + " record";

var initValues = new Array();
initValues.schedtype = SCHEDULE_TYPE;
initValues.project = P1;

//Create record
var myBillingSched = record.create(
    type: record.Type.BILLING_SCHEDULE,
    isDynamic: true,
    defaultValues: initValues
);

myBillingSched.setValue(
    fieldId: 'name',
    value: recName
);
myBillingSched.setValue(
    fieldId: 'initialamount',
    value: '10'    // 10%
);

//Create the sublist
myBillingSched.selectNewLine({
    sublistId: 'milestone'
});
myBillingSched.setCurrentSublistValue({
    sublistId: 'milestone',
    fieldId: 'milestoneamount',
    value: '25'    // 25%
});
myBillingSched.setCurrentSublistValue({
    sublistId: 'milestone',
    fieldId: 'milestonedate',
    value: new Date('November 21, 2013')
});
myBillingSched.setCurrentSublistValue({
    sublistId: 'milestone',
    fieldId: 'projecttask',
    value: P1M1
});
```
myBillingSched.commitLine({
  sublistId: 'milestone'
});
recId = myBillingSched.save();

//Update the project (link it with the newly created Billing Schedule)
var project = record.load({
  type: record.Type.JOB,
  id: P1
});
project.setValue({
  fieldId: 'jobbillingtype',
  value: SCHEDULE_TYPE
});
project.setValue({
  fieldId: 'billingschedule',
  value: recId
});
var pId = project.save();

Bin

**Note:** This topic applies to all versions of SuiteScript.

Bins help you track on-hand quantities within a warehouse. Tracking items by bins can help organize receiving items and simplify picking items to fulfill orders.

For example, if you use bins, when you receive a purchase order, the order can tell you which bin to put the items away in. Then, your stock level of that item in that bin is tracked.

To use the bin record, the Bin Management feature must be enabled at Setup > Company > Setup Tasks > Enable Features. On the Items & Inventory tab, check the Bin Management box.

For help working with this record in the UI, see the help topic Bin Management.

The internal ID for this record is bin.

See the [SuiteScript Records Browser](https://www.netsuite.com) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

Client SuiteScript is not supported for the bin record. It is scriptable in server SuiteScript only.

**Supported Functions**

The bin record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
A Bill of Materials, or BOM, lists the quantities of raw materials, assemblies, sub-components, and parts needed to manufacture a product. A BOM can be used to communicate between manufacturing partners, multiple facilities within the organization, or with a single manufacturing plant. The BOM that is scriptable in NetSuite is the Advanced BOM.

This record is supported when the Advanced BOM feature is enabled at Setup > Company > Enable Features, on the Items & Inventory tab. After this feature has been enabled, the Advanced BOM replaces the Assembly/Bill of Materials record.

For help working with this record in the UI, see the help topic Advanced Bill of Materials.

In the UI, this record is accessed by going to Lists > Supply Chain > Bill of Materials.

The internal ID for this record is `bom`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The BOM record is scriptable in client and server SuiteScript.

### Supported Functions

The BOM record is fully scriptable, which means that it can be read, created, edited, copied, deleted, and searched using SuiteScript.

### BOM Code Samples

- **Create a BOM**
- **Copy a BOM**
- **Delete a BOM**
- **Full BOM Code Sample**

#### Create a BOM

The following sample shows how to create a new bill of materials record.

```javascript
require(['N/record'], function(record){
  function createBom(values){
    var rec = record.create({
      type: record.Type.BOM,
      isDynamic: true
    });
  }
});
```
Copy a BOM

The following sample shows how to copy a bill of materials record.

```javascript
require(['N/record'], function(record) {
    function copyBom(bomId, values) {
        var rec = record.copy({
            type: record.Type.BOM,
            id: bomId,
            isDynamic: true
        });
        for (var key in values) {
            if (values.hasOwnProperty(key)) {
                rec.setValue({
                    fieldId: key,
                    value: values[key]
                });
            }
        }
        return rec.save({enableSourcing: false, ignoreMandatoryFields: false});
    }
    var bomId = createBom(
        'name': 'SS 2.0 BoM',
        'subsidiary': [4, 5],
        'includechildren': true,
        'usecomponentyield': true
    );
});
```

Delete a BOM

The following sample shows how to delete a bill of materials record.

```javascript
require(['N/record'], function(record) {
    function deleteBom(bomId) {
        record.delete({
            id: bomId,
            isDynamic: true
        });
    }
    var bomId = createBom(
        'name': 'SS 2.0 BoM Copy',
        'subsidiary': [4],
        'includechildren': false,
        'usecomponentyield': false
    );
});
```
Lists

276
type: record.Type.BOM,
id: bomId
});
deleteBom2();
});

Full BOM Code Sample

The following sample shows how to create, copy, and delete a bill of materials record, including edits to BOM revisions. For samples specific to BOM revisions, see BOM Revision Code Samples.

```
require(['N/record', 'N/search'], function(record, search){
// create bill of materials (function)
function createBom(values){
    var rec = record.create({
        type: 'bom',
        isDynamic: true
    });
    for (var key in values){
        if (values.hasOwnProperty(key)){
            rec.setValue({
                fieldId: key,
                value: values[key]
            });
        }
    }
    return rec.save({
        enableSourcing: false,
        ignoreMandatoryFields: false
    });
}

// create bill of materials revision (function)
function createBomRevision(values, components){
    var rec = record.create({
        type: 'bomrevision',
        isDynamic: true
    });
    for (var key in values){
        if (values.hasOwnProperty(key)){
            rec.setValue({
                fieldId: key,
                value: values[key]
            });
        }
    }
    for (var i in components){
        rec.selectNewLine({
            sublistId: 'component'
        });
        for (var key in components[i]){  
            if (components[i].hasOwnProperty(key)){
                rec.setCurrentSublistValue({
                    sublistId: 'component',
                    fieldId: key,
                    value: values[key]
                });
            }
        }
    }
}
```

SuiteScript Records Guide
Lists

```
fieldId: key,
   value: components[i][key]
 });
}
rec.commitLine({sublistId: 'component'});
return rec.save({enableSourcing: false, ignoreMandatoryFields: false});
}

// update bill of materials revision (function)
function updateBomRevision(revisionId, values){
  var rec = record.load({
    type: 'bomrevision',
    id: revisionId,
    isDynamic: true
  });
  for (var key in values){
    if (values.hasOwnProperty(key)){
      rec.setValue({
        fieldId: key,
        value: values[key]
      });
    }
  }
  return rec.save({enableSourcing: false, ignoreMandatoryFields: false});
}

// copy bill of materials (function)
function copyBom(bomId, values){
  var rec = record.copy({
    type: 'bom',
    id: bomId
  });
  for (var key in values){
    if (values.hasOwnProperty(key)){
      rec.setValue({
        fieldId: key,
        value: values[key]
      });
    }
  }
  return rec.save({enableSourcing: false, ignoreMandatoryFields: false});
}

// delete bill of materials (function)
function deleteBom(bomId){
  record.delete(
    {type: 'bom',
     id: bomId
  });
}

// get bill of materials revisions (function)
function getBomRevisions(bomId){
```
// search ids for all revisions for given BoM
var searchObj = search.create(
    type: 'bomrevision',
    filters: [
        ['billofmaterials', 'is', [bomId]],
    ],
    columns: ['name']
);
var revisions = [];
searchObj.run().each(function(result){
    revisions.push(
        {id: result.id,
         name: result.getValue('name')
        });
    return true;
});
return revisions;

// create BoM - call createBom function
var bom1Id = createBom({
    name: 'SS 2.0 BoM',
    subsidiary: [4,5],
    includechildren: true,
    usecomponentyield: true
});

// create new BoM revision - call createBomRevision function
var bom1Rev1Id = createBomRevision({
    billofmaterials: bom1Id,
    name: 'Rev.1',
    effectivestartdate: new Date('1/1/2017')
}, [  
    {'item': 63, 'bomquantity': 1},
    {'item': 64, 'bomquantity': 2}
]);

// update BoM Revision Effective End Date to prevent a collision - call updateBomRevision function
bom1Rev1Id = updateBomRevision(bom1Rev1Id, {
    effectiveenddate: new Date('12/31/2018')
});

// create another new BoM revision - call createBomRevision function
var bom1Rev2Id = createBomRevision({
    billofmaterials: bom1Id,
    name: 'Rev.2',
    effectivestartdate: new Date('1/1/2019')
}, [  
    {'item': 64, 'bomquantity': 3},
    {'item': 62, 'bomquantity': 1}
]);

// copy BoM and update some values - call copyBom function */
var bom2Id = copyBom(bom1Id, {
    name: 'SS 2.0 BoM Copy',
});
```javascript

'subsidiary': [4],
'usecomponentyield': false,
'includedchildren': false
});

// find BoM Revisions of BoM - call getBomRevisions function
var revisions = getBomRevisions(bom1Id);

// update first component's Component Yield value of each found BoM Revision
for (var i = 0; i < revisions.length; i++) {
  rec = record.load({
    type: 'bomrevision',
    id: revisions[i].id
  });
  rec.setSublistValue({
    sublistId: 'component',
    fieldId: 'componentyield',
    line: 0,
    value: 50
  });
  rec.save();
}

// delete BoMs - call deleteBom function
deleteBom(bom1Id);
delteBom(bom2Id);
```

**BOM Revision**

---

**Note:** This topic applies to all versions of SuiteScript.

A BOM revision enables you to update a BOM's details throughout the product lifecycle. A BOM revision also provides an accessible revisions history. Use revisions to compare and track cost savings when many BOM revisions are used in production. This record is available when the Advanced BOM feature is enabled at Setup > Company > Enable Features, on the Items & Inventory tab. For information about this feature, see the help topic Advanced Bill of Materials.

In the UI, this record is accessed by going to Lists > Supply Chain > Bill of Materials.

For help working with this record in the UI, see BOM Revision.

The internal ID for this record is **bomrevision**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

---

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types

The BOM Revision record is scriptable in server SuiteScript only.

Supported Functions

The BOM Revision record is fully scriptable, which means that it can be read, created, edited, copied, deleted, and searched using SuiteScript.

BOM Revision Code Samples

- Create a BOM Revision
- Search for a BOM Revision
- Update a BOM Revision
- Full BOM Code Sample

Create a BOM Revision

The following sample shows how to create a new bill of materials revision record.

```javascript
require(['N/record'], function(record){
    function createBomRevision(values, items){
        var rec = record.create({
            type: 'bomrevision', isDynamic: true
        });
        for (var key in values){
            if (values.hasOwnProperty(key)){
                rec.setValue({
                    fieldId: key,
                    value: values[key]
                });
            }
        }
        for (var itemIndex in items){
            rec.selectNewLine({
                sublistId: 'component'
            });
            for (var key in items[itemIndex]){  
                if (items[itemIndex].hasOwnProperty(key)){
                    rec.setCurrentSublistValue({
                        sublistId: 'component',
                        fieldId: key,
                        value: items[itemIndex][key]
                    });
                }
            }
            rec.commitLine({sublistId: 'component'});
        }
        return rec.save({enableSourcing: false, ignoreMandatoryFields: false});
    }
    var bomRevisionId = createBomRevision({
```
Search for a BOM Revision

The following sample shows how to search for a bills of materials revision record.

```
require(['N/search'], function(search){
  function getBomRevisions(bomId){
    // search ids for all revisions for given BoM
    var searchObj = search.create({
      type: 'bomrevision',
      filters: [{
        name: 'billofmaterials',
        operator: search.Operator.IS,
        values: [bomId]
      }],
      columns: ['name']
    });
    var revisions = [];
    searchObj.run().each(function(result) {
      revisions.push({
        id: result.id,
        name: result.getValue('name')
      });
      return true;
    });
    return revisions;
  }
  var revisions = getBomRevisions(1);
});
```

Update a BOM Revision

The following sample shows how to update a bills of materials revision record.

```
require(['N/record'], function(record){
  function updateBomRevision(revisionId, values){
    var rec = record.load({
      type: 'bomrevision',
      id: revisionId
    });
    for (var key in values){
      if (values.hasOwnProperty(key)){
        recsetValue({
          fieldId: key,
          value: values[key]
        });
      }
    }
  }
});
```
Budget Exchange Rate

**Note:** This topic applies to all versions of SuiteScript.

The Budget Exchange Rates table is available at Lists > Accounting > Budget Exchange Rates. In this table you can maintain exchange rates between the child and parent subsidiaries for use in the budgeting process.

**Note:** To access this table, the Multiple Currencies and Multiple Budgets features must be enabled.

The Budget Exchange Rates table is very similar to the Consolidated Exchange Rates table. In NetSuite OneWorld, NetSuite uses the Consolidated Exchange Rates table to initially populate the Budget Exchange Rates table. Then, you can update and maintain the Budget Exchange Rates table as necessary.

This table shows the exchange rates for each subsidiary's base currency in relation to the currencies used by its parent subsidiaries, for each period. Table columns include: accounting period, accounting book, whether the period is closed, the From (child) subsidiary, the To (parent) subsidiary, and exchange rates.

Rates in this table are either direct or indirect (derived). Direct rates are rates set between a child and parent subsidiary. Direct rates are set by a user and may be edited. Indirect rates are rates set between a grandchild and grandparent subsidiary. Indirect rates are always calculated by the system and cannot be edited. Budget exchange rates include three different rate types per period, subsidiary, and accounting book: Average, Current, and Historical.

For more information about the budget exchange rate record and working with it in the UI, see the help topic Budget Exchange Rates.

The internal ID for this record is `budgetexchangerate`.

The budget exchange rate record is available only when the Multiple Currencies feature is enabled. An administrator can enable this feature at Setup > Company > EnableFeatures > Company subtab.

The script user must have the Currency permission with Full permission level. To edit the different rate types per period and subsidiary pair, the script user must also have access to both the From and To subsidiary.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
Supported Script Types

The budget exchange rate record is scriptable in both client and server SuiteScript. The editable elements of the record are the currentRate, averageRate, and historicalRate. The remaining record elements can be read and searched.

**Important:** Rates are created and deleted when subsidiaries or secondary accounting books are created and deleted. Script users can edit the record only if it is not derived (Element field-> isDerived) and only when the accounting period to which this rate belongs is not closed (Element field-> isPeriodClosed). In addition, the script user cannot edit a budget exchange rate of 1 for an elimination subsidiary and its direct parent (Element field-> isEliminationSubsidiary), with the following exception. Those customers who already have a non-1 budget exchange rate (current, average, or historical) between an elimination subsidiary and its direct parent subsidiary can edit the budget exchange rates for the elimination subsidiary.

Supported Functions

This record is partially scriptable. It can be read, updated, and searched using SuiteScript. It cannot be created, copied, deleted, or transformed.

Code Samples

The following sample shows how to update a budget exchange rate between two subsidiaries.

```javascript
var PARENT_SUBSIDIARY = 1;
var CHILD_SUBSIDIARY = 3;
var PERIOD = 212;

// Search for the budget rate record from CHILD_SUBSIDIARY to PARENT_SUBSIDIARY for a specific PERIOD
var mySearch = search.create({
  type: 'budgetexchangerate',
  filters: [
    { name: 'fromsubsidiary',
      operator: search.Operator.IS,
      values: CHILD_SUBSIDIARY
    },
    { name: 'tosubsidiary',
      operator: search.Operator.IS,
      values: PARENT_SUBSIDIARY
    },
    { name: 'period',
      operator: search.Operator.IS,
      values: PERIOD
    }
  ],
  columns: [{
    name: 'internalid'
  }]
});
var searchResults = mySearch.run().each(function(result){
```
var fromSubsidiary = result.getValue({
    name: 'fromsubsidiary'
});
var toSubsidiary = result.getValue({
    name: 'tosubsidiary'
});
var period = result.getValue({
    name: 'period'
});
return true;

// Update the budget rate's current, average, and historical rate
var id1 = searchResults[0].getValue({
    name: 'internalid'
});
var rateRecord = record.load({
    type: record.Type.BUDGET_EXCHANGE_RATE,
    id: id1,
    defaultValues: true
});
rateRecord.setValue({
    fieldId: 'currentrate',
    value: 1.2
});
rateRecord.setValue({
    fieldId: 'averagerate',
    value: 1.3
});
rateRecord.setValue({
    fieldId: 'historicalrate',
    value: 1.4
});
recId = rateRecord.save();

Class

Note: This topic applies to all versions of SuiteScript.

Classes are categories that you can create to track records such as financials, transactions, and employees. For example, you own a janitorial service and you want to track income and expenses separately for household and commercial accounts. You set up a class for each account type and track the financial performance of each class over any time period.

For help working with this record in the UI, see , see the help topic Creating Classes.

The internal ID for this record is classification.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types

Client SuiteScript is not supported for this record. The class record is scriptable in server SuiteScript.

Supported Functions

The class record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Consolidated Exchange Rate

**Note:** This topic applies to all versions of SuiteScript.

If you use NetSuite OneWorld and your subsidiaries have different base currencies, you can maintain a table of consolidated exchange rates. This table is used to ensure that for consolidation purposes, currency amounts properly roll up from child to parent subsidiaries. Each consolidated exchange rate translates between the base currency of a subsidiary and the base currency of its parent or grandparent subsidiary, for a specified accounting period. Consolidated exchange rates include three different rate types per period, subsidiary, and accounting book: Current, Average, and Historical.

The internal ID for this record is `consolidatedexchangerate`.

The consolidated exchange rate record is available only when the Multiple Currencies feature is enabled. An administrator can enable this feature at Setup > Company > EnableFeatures > Company subtab.

For help working with this record in the UI, see the help topic [Consolidated Exchange Rates](#).

The script user must have the Currency permission with Full permission level. To edit the different rate types per period and subsidiary pair, the script user must also have access to both the From and To subsidiary.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The consolidated exchange rate record is scriptable in both client and server SuiteScript.

The editable elements of the record are the currentrate, averagerate, historicalrate, and externalid. The remaining record elements can be read and searched.
Important: Rates are created and deleted when subsidiaries or secondary accounting books are created and deleted. Script users can edit the record only if it is not derived (Element field-> isderived) and only when the accounting period to which this rate belongs is not closed (Element field-> isperiodclosed). In addition, the script user cannot edit a consolidated exchange rate of 1 for an elimination subsidiary and its direct parent (Element field-> iseliminationsubsidiary), with the following exception. Those customers who already have a non-1 consolidated exchange rate (current, average, or historical) between and elimination subsidiary and its direct parent subsidiary can edit the consolidated exchange rates for the elimination subsidiary.

Supported Functions

This record is partially scriptable. It can be read, updated, and searched using SuiteScript. It cannot be created, deleted, or transformed.

Code Samples

The following sample show how to update a consolidated rate between two subsidiaries.

```javascript
var PARENT_SUBSIDIARY = 1;
var CHILD_SUBSIDIARY = 3;
var PERIOD = 212;

// Search for the consolidated rate record from CHILD_SUBSIDIARY to PARENT_SUBSIDIARY for a specific PERIOD
var mySearch = search.create(
    type: ‘consolidatedexchangerate’,
    filters: [{
        name: ‘fromsubsidiary’,
        operator: search.Operator.IS,
        values: CHILD_SUBSIDIARY
    }, {
        name: ‘tosubsidiary’,
        operator: search.Operator.IS,
        values: PARENT_SUBSIDIARY
    }, {
        name: ‘period’,
        operator: search.Operator.IS,
        values: PERIOD
    }],
    columns: [{
        name: ‘internalid’
    }],
);
mySearch.run().each(function(result){
    var fromSubsidiary = result.getValue({
        name: ‘fromsubsidiary’
    });
    var toSubsidiary = result.getValue({
        name: ‘tosubsidiary’
    });
    var period = result.getValue({
        name: ‘period’
    });
    return true;
});
```
Update the consolidated rate's current, average, and historical rate

```javascript
var rateRecord = record.load(
    { 
        type: record.Type.CONSOLIDATED_EXCHANGE_RATE, 
        id: searchResults[0].getValue('internalid'), 
        defaultValues: true 
    });
rateRecord.setValue(
    { 
        fieldId: 'currentrate',
        value: 1.2
    });
rateRecord.setValue(
    { 
        fieldId: 'averagerate',
        value: 1.3
    });
rateRecord.setValue(
    { 
        fieldId: 'historicalrate',
        value: 1.4
    });
var recId = rateRecord.save();
```

**Contact Category**

ℹ️ **Note:** This topic applies to all versions of SuiteScript.

Contact category defines a list of values that are used by the contact record to set the type of contact. In the UI, this is a user defined list at Setup > Sales > CRM Lists > Contact Category.

For help working with this record in the UI, see the help topic Setting Up Accounting Lists.

The internal ID for this record is `contactcategory`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

ℹ️ **Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The contact category record is scriptable in both client and server SuiteScript.

**Supported Functions**

The contact category record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Contact Role

**Note:** This topic applies to all versions of SuiteScript.

Contact roles can be defined for each contact listed on various records including opportunity, customer, vendor, and partner.

For help working with this record in the UI, see the help topic Contacts.

The internal ID for this record is `contactrole`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The contact role record is scriptable in both client and server SuiteScript.

**Supported Functions**

The contact role record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

---

Cost Category

**Note:** This topic applies to all versions of SuiteScript.

Cost category records are used to classify different types of costs associated with your items. Using cost categories helps you to track costs and variances in the manufacturing process. These categories are available when the Standard Costing or Landed Cost feature is enabled.

Material or service cost categories track standard costs for items.

For help working with this record in the UI, see the help topic Creating Cost Categories.

Landed cost category values are used for items on bills and item receipts to categorize the different kinds of expenses incurred when making purchases. For details, see Landed Cost Categories.

The internal ID for this record is `costcategory`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types

The cost category record is scriptable in both client and server SuiteScript.

Supported Script Functions

The cost category record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Currency

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Currency Management.

The internal ID for this record is `currency`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The currency record is scriptable in server SuiteScript only.

Supported Functions

When the Multiple Currencies feature is enabled, full server-side scripting is supported for this record type.

When the Multiple Currencies feature is not enabled, scripting does not support create, edit, delete, or search of currency records. Search-based functions such as `search.lookupFields(options)` are not supported, because search is not supported. Loading of a currency record to get field values is supported, as long as the currency ID is known, as shown in the following sample:

```javascript
var rec = record.load({
  type: record.Type.CURRENCY,
  id: 1
});
var symbol=rec.getValue({
```

---

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Currency Management.

The internal ID for this record is `currency`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The currency record is scriptable in server SuiteScript only.

Supported Functions

When the Multiple Currencies feature is enabled, full server-side scripting is supported for this record type.

When the Multiple Currencies feature is not enabled, scripting does not support create, edit, delete, or search of currency records. Search-based functions such as `search.lookupFields(options)` are not supported, because search is not supported. Loading of a currency record to get field values is supported, as long as the currency ID is known, as shown in the following sample:

```javascript
var rec = record.load({
  type: record.Type.CURRENCY,
  id: 1
});
var symbol=rec.getValue({
```
Usage Notes

As of 2016.1, the `currencyprecision` field is available for scripting, even when the `ALLOWCURRENCYPRECISIONCHANGE` preference is disabled and the field is read-only. In scripting, this field is always a number and never a list box, behavior which differs from this field's behavior in the UI.

Customer Category

**Note:** This topic applies to all versions of SuiteScript.

Customer category defines a list of values that are used by the customer record to set the type of customer. In the UI, reference an existing customer category list by going to Setup > Accounting > Accounting Lists.

To create a new customer category list, set the Type list at the top of the page to Customer Category. Next, click the New button that appears at the top of the page.

For help working with this record in the UI, see the help topic Customers.

The internal ID for this record is `customercategory`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The customer category record is scriptable in both client and server SuiteScript.

Supported Functions

The customer category record is partially scriptable — it can be created, updated, copied, and deleted using SuiteScript. Search is not available for the customer category record.

Customer Message

**Note:** This topic applies to all versions of SuiteScript.

Standardized customer messages can be created to be available for selection on transaction records. A selected message is sent to the customer for the transaction. For example, a message of "Thank you for your order!" can be selected on an invoice record to be displayed on the invoice sent to the customer.
For help working with this record in the UI, see the help topic Setting Up Accounting Lists.

The internal ID for this record is **customermessage**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The customer message record is scriptable in both client and server SuiteScript.

**Supported Functions**

The customer message record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Customer-Subsidiary Relationship**

**Note:** This topic applies to all versions of SuiteScript.

If you use NetSuite OneWorld and have enabled the Multi-Subsidiary Customer feature, you can share the customer and sub-customer records with multiple subsidiaries, and then select those subsidiaries on core transactions. An administrator can enable these features at Setup > Company > EnableFeatures, on the Company subtab.

For help working with Multi-Subsidiary Customers in the UI, see the help topic Assigning Subsidiaries to a Customer.

The internal ID for this record is **customersubsidiaryrelationship**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For customer-subsidiary and vendor-subsidiary relationships belonging to the same entity, both share the external ID of the entity. For example, Company ABC can be both a vendor and customer. If Company ABC has a vendor and customer relationship with the US subsidiary, both share the same external ID.

**Supported Script Types**

The customer-subsidiary relationship is scriptable in both client and server SuiteScript.
All three user events are supported: **beforeLoad**, **beforeSubmit**, and **afterSubmit**.

## Supported Functions

The customer-subsidiary relationship can be read, created, updated, and deleted.

### Code Sample

The following sample shows how to create a customer-subsidiary relationship.

```javascript
var csr = record.create({
    type: record.Type.CUSTOMER_SUBSIDIARY_RELATIONSHIP
});
csr.setValue(
    {
        fieldId: 'entity',
        value: 42
    }
);
csr.setValue(
    {
        fieldId: 'subsidiary',
        value: 1
    }
);
var recId = csr.save();
```

## Department

**Note:** This topic applies to all versions of SuiteScript.

Departments are categories that you can create to separate and track records such as financials, transactions, and employees. For example, you can create a department for each team of employees dedicated to a certain area of business, and then track income and expenses by each department over any time period.

For help working with this record in the UI, see the help topic **Creating Departments**.

The internal ID for this record is `department`.

See the **SuiteScript Records Browser** for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic **Working with the SuiteScript Records Browser** in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

## Supported Script Types

Client SuiteScript is not supported for this record. The department record is scriptable in server SuiteScript.
Supported Functions

The department record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Employee Change Request

**Important:** This topic describes a feature that is currently available only in some customer accounts. For information on the availability of this feature for your account, please contact your NetSuite account manager.

Employee Change Requests allow managers to initiate requests for changes to their employees in NetSuite. Employee change request types must be set up and assigned to an employee change reason. Managers can initiate an employee change request for their direct reports in the Employee Center and have it go through an approval process which you can define in SuiteFlow. Approvers can approve or decline employee change requests and approved requests will be reflected in the employee’s record on a proposed future date. For more information about employee change requests, see the help topic Employee Change Requests.

The internal ID for this record is `employeechangerequest`.

See the [SuiteScript Records Browser](https://www.netsuite.com) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](https://www.netsuite.com) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](https://www.netsuite.com)
- [N/record Module](https://www.netsuite.com)

Prerequisites

To use the employee change request record, you must enable and set up the Employee Change Requests, Effective Dating, and Advanced Employee Permissions features at Setup > Company > Enable Features, on the Employees tab. To create an employee change request approval workflow in SuiteFlow, you must also enable the SuiteFlow feature at Setup > Company > Enable Features, on the SuiteCloud tab.

Supported Script Types

The employee change request record is scriptable in both client and server SuiteScript.

Field IDs

<table>
<thead>
<tr>
<th>Field Label</th>
<th>Internal ID</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Justification</td>
<td>justification</td>
<td>text</td>
</tr>
<tr>
<td>Requester</td>
<td>requester</td>
<td>select</td>
</tr>
<tr>
<td>Proposed Date</td>
<td>proposeddate</td>
<td>date</td>
</tr>
<tr>
<td>Next Approver</td>
<td>nextapprover</td>
<td>select</td>
</tr>
<tr>
<td>Field Label</td>
<td>Internal ID</td>
<td>Type</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Request Status</td>
<td>requeststatus</td>
<td>select</td>
</tr>
<tr>
<td>Inactive</td>
<td>isinactive</td>
<td>checkbox</td>
</tr>
<tr>
<td>Change Type</td>
<td>employeechangerequesttype</td>
<td>select</td>
</tr>
<tr>
<td>Declining Reason</td>
<td>decliningreason</td>
<td>text</td>
</tr>
<tr>
<td>Declined By</td>
<td>declinedby</td>
<td>select</td>
</tr>
</tbody>
</table>

**Supported Functions**

The following SuiteScript functionality is supported:

- Read
- Create
- Update
- Delete
- Search

Copy and transform are not supported.

**Usage Notes**

The following fields cannot be edited and are only available on create:

- requester
- employee
- <requested fields>

**Code Samples**

> **Note:** The proposeddate field syntax must match the format you have chosen in your company settings. Go to Setup > Company > Preferences > General Preferences to configure your Date Format.

The following sample shows how to create an employee change request.

```javascript
require(['N/record'], function(record){
    var empChangeReq = record.create({
        type : record.Type.EMPLOYEE_CHANGE_REQUEST
    });
    empChangeReq.setValue({
        fieldId : 'employeechangerequesttype',
        value: '1'
    });
    empChangeReq.setText({
        fieldId : 'proposeddate',
        text: '10/10/2020'
    });
});
```
Employee Change Request Type

**Important:** This topic describes a feature that is currently available only in some customer accounts. For information on the availability of this feature for your account, please contact your NetSuite account manager.

Employee change request types provide managers with the types of changes they can initiate for their direct reports. For more information about employee change request type, see the help topic Setting Up Employee Change Request Types.

The internal ID for this record is `employeechangerequesttype`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- N/record Module

**Prerequisites**

To use the employee change request type record, you must enable and set up the Employee Change Requests, Effective Dating, and Advanced Employee Permissions features at Setup > Company > Enable Features, on the Employees tab.

**Supported Script Types**

The employee change request type record is scriptable in both client and server SuiteScript.

**Field IDs**

<table>
<thead>
<tr>
<th>Field Label</th>
<th>Internal ID</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>name</td>
<td>text</td>
</tr>
<tr>
<td>Description</td>
<td>description</td>
<td>text</td>
</tr>
<tr>
<td>Change Reason</td>
<td>changereason</td>
<td>select</td>
</tr>
</tbody>
</table>
Supported Functions

The following SuiteScript functionality is supported:

- Read
- Create
- Update
- Delete
- Search

Copy and transform are not supported.

Code Sample

The following code sample shows how to create an employee change request type.

```javascript
require(['N/record'], function(record){
    var empChangeReqType = record.create({
        type : record.Type.EMPLOYEE_CHANGE_REQUEST_TYPE
    });
    empChangeReqType.setValue({
        fieldId : 'name',
        value: 'Test 2'
    });
    empChangeReqType.setValue({
        fieldId : 'description',
        value: 'Test'
    });
    empChangeReqType.setValue({
        fieldId : 'changereason',
        value: "10"
    });
    empChangeReqType.save();
});
```

Expense Category

**Note:** This topic applies to all versions of SuiteScript.

Expense categories are used to group expenses. Popular categories include transportation, lodging, mileage, and entertainment. Each expense category is linked to an account. When an employee enters an expense report, he or she selects a category for each expense, and the expense automatically posts to the associated expense account. Note that new expense categories cannot be created at the time an expense report is entered.

If you use NetSuite OneWorld, be aware that an expense category is available to only those subsidiaries assigned to the account linked with the expense category. If you want to enable intercompany expense transactions, it is recommended that you set up expense categories linked to expense accounts that are available to all subsidiaries, for use in these transactions. For intercompany expense transactions, users cannot save expense lines unless they contain expense categories available to both the employee subsidiary and customer subsidiary.
To enter expense categories, go to Setup > Accounting > Setup Tasks > Expense Categories > New (Administrator).

For help working with this record in the UI, see the help topic Creating an Expense Category.

The internal ID for this record is `expensecategory`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The expense category record is scriptable in server SuiteScript only.

**Supported Functions**

The expense category record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Fair Value Price**

**Note:** This topic applies to all versions of SuiteScript.

Fair Value Price is a list of the records that define the fair value for items. Fair value price is used to allocate revenue in revenue arrangements.

This record is part of advanced revenue management. To use advanced revenue management, the Accounting Periods feature and Advanced Revenue Management feature must be enabled. Before you begin working with advanced revenue management programmatically, see the help topic Setup for Advanced Revenue Management.

For help working with this record in the UI, see the help topic Fair Value Setup.

The internal ID for this record is `fairvaluepricelist`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
N/record Module

**Supported Script Types**

The fair value price record is scriptable in server SuiteScript only.

**Supported Functions**

The fair value price record is partially scriptable. It can be created, deleted, and searched using SuiteScript. You cannot copy the fair value price record.

**Financial Institution**

- **Note:** This topic applies to all versions of SuiteScript.

The financial institution record allows you to save multiple format profiles for a financial institution.

For help working with this record in the UI, see the help topic Financial Institution Records.

The internal ID for this record is financialinstitution.

This record is available in NetSuite and NetSuite OneWorld accounts.

The script user must have the Financial Institution Records permission with Full Access.

In the UI, you create financial institution records on the Financial Institution page at Setup > Accounting > Financial Institution.

See the SuiteScript Records Browser for all internal IDs associated with this record.

- **Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is scriptable in server SuiteScript. This record does not support client SuiteScript.

**Supported Functions**

This record is partially scriptable — it can be read, created, updated, deleted, and searched. It cannot be copied or transformed.

**Usage Notes**

The table on the Financial Institution page is not editable or scriptable. Clicking **Add** opens the Format Profile page.
Code Sample

The following sample shows how to create a financial institution record.

```javascript
// Create a financial institution record
var newFIRecord = record.create({
    type: record.Type.FINANCIAL_INSTITUTION,
    isDynamic: false,
    defaultValues: null
}).setValue({
    fieldId: 'financialinstitution',
    value: 'New XYZ Bank'
}).setValue({
    fieldId: 'description',
    value: 'Description for XYZ Bank'
}).save();

// Load a financial institution record
var createdFIRecord = record.load({
    type: record.Type.FINANCIAL_INSTITUTION,
    id: 1,
    isDynamic: false
});

// Get value of the description field
var description = createdFIRecord.getValue({
    fieldId: 'description'
});

// Update the description field
createdFIRecord.setValue({
    fieldId: 'description',
    value: 'Edited Description for XYZ Bank'
});
var recId = createdFIRecord.save();
```

Format Profile

**Note:** This topic applies to all versions of SuiteScript.

Format profile records enable you to set up account mappings and custom bank transaction codes for statement file formats. A format profile can only be associated with one financial institution record. If you delete a financial institution record, NetSuite deletes the associated format profile as well provided it does not have any associated imports.

For help working with this record in the UI, see the help topic Creating Format Profiles.

The internal ID for this record is `formatprofile`.

This record is available in NetSuite and NetSuite OneWorld accounts.

The script user must have the Financial Institution Records permission with Full Access.

In the UI, you create a format profile record on the Format Profile page, which you get to by going to Setup > Accounting > Financial Institution and clicking Add.
See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

The format profile record has two sublists:
- Account Mapping
- Transaction Code Mapping

**Account Mapping**

The Account Mapping sublist allows you to map accounts in the imported bank statements to corresponding NetSuite accounts.

For help working in this sublist in the UI, see the help topic Bank Account Mapping.

The internal ID for this sublist is `accountmapping`.

This sublist is available in NetSuite and NetSuite OneWorld accounts.

In the UI, you create an Account Mapping sublist on the Format Profile page by clicking Add in the Account Mapping subtab. To get to the Format Profile page, go to Setup > Accounting > Financial Institution, and then click Add.

**Transaction Code Mapping**

The Transaction Code Mapping sublist allows you to map custom bank transaction codes to corresponding bank data types in format profile records.

For help working in this sublist in the UI, see the help topic Mapping Custom Bank Codes.

The internal ID for this sublist is `trancodemapping`.

This sublist is available in NetSuite and NetSuite OneWorld accounts.

In the UI, you create a Transaction Code Mapping sublist on the Format Profile page by clicking Add in the Transaction Code Mapping subtab. To get to the Format Profile page, go to Setup > Accounting > Financial Institution, and then click Add.

**Supported Script Types**

The format profile record and its sublists are scriptable in both server and client SuiteScript.

**Supported Functions**

The format profile record and its sublists are partially scriptable — they can be read, created, updated, and deleted. They cannot be copied, transformed, or searched.
Usage Notes

In the UI, the Financial Institution field is auto-populated on the Format Profile page and cannot be changed. In SuiteScript, users can change this field.

Code Sample

The following sample shows how to create a format profile record.

```javascript
// Create a format profile record
var newFPRecord = record.create({
    type: record.Type.FORMAT_PROFILE,
    isDynamic: true
});
newFPRecord.setValue({
    fieldId: 'formatprofile',
    value: 'XYZ'
});
newFPRecord.setValue({
    fieldId: 'financialinstitution',
    value: 2
});
newFPRecord.setValue({
    fieldId: 'transactionparser',
    value: 201
});
newFPRecord.setValue({
    fieldId: 'description',
    value: 'Description for XYZ Format Profile'
});
newFPRecord.selectNewLine({
    sublistId: 'accountmapping'
});
newFPRecord.setCurrentSublistValue({
    sublistId: "accountmapping",
    fieldId: "accountmappingkey",
    value: 123
});
newFPRecord.setCurrentSublistValue({
    sublistId: 'accountmapping',
    fieldId: 'mappednetsuiteaccount',
    value: 1
});
newFPRecord.commitLine({
    sublistId: 'accountmapping'
});
var fpID = newFPRecord.save();

// Load a format profile record
var createdFPRecord = record.load({
    type: record.Type.FORMAT_PROFILE,
    id: fpID,
    isDynamic: true
});
```
Gift Certificate

Note: This topic applies to all versions of SuiteScript.

You can create gift certificate items that allow customers to purchase store credit they can send to someone as a gift. The recipient uses the gift certificate code when placing an order through your Web store or entering a transaction with a sales representative. You can set a preference in NetSuite for how you want to generate the gift certificate codes: you can create them yourself, or use a random hash code automatically generated by the system.

Gift certificate codes are not active until the order used to purchase the gift certificate is billed.

For help working with this record in the UI, see the help topic Gift Certificates.

The internal ID for this record is giftcertificate.
See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The gift certificate record is scriptable in both client and server SuiteScript.

**Supported Functions**

The gift certificate record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

**Global Account Mapping**

**Note:** This topic applies to all versions of SuiteScript.

For accounts using Multi-Book Accounting, the global account mapping record enables you to configure secondary accounting books to post to accounts different from the primary book. These mappings are used by transactions where the user can manually select the account to which the transaction posts.

The internal ID for this record is `globalaccountmapping`.

Using this record requires that, as part of your Multi-Book configuration, you select the Chart of Accounts Mapping option at Setup > Enable Features, on the Accounting subtab, in addition to the other setup steps required for Multi-Book Accounting.

In the UI, you access this record at Setup > Accounting > Global Account Mappings > New.

For help working with this record in the UI, see the help topic Global Account Mapping.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is scriptable in server SuiteScript only.

All three user events are supported: `beforeLoad`, `beforeSubmit`, and `afterSubmit`.
Supported Functions

This record is fully scriptable, which means it can be created, updated, copied, deleted, and searched using SuiteScript.

Code Samples

The following samples show how to create global account mapping records and perform other basic tasks.

```javascript
// Create Record
var gamRec = record.create({
  type: record.Type.GLOBAL_ACCOUNT_MAPPING
});
gamRec.setValue({
  fieldId: 'effectivedate',
  value: new Date('4/4/2004')
});
gamRec.setValue({
  fieldId: 'accountingbook',
  value: '2'
});
gamRec.setValue({
  fieldId: 'sourceaccount',
  value: '6'
});
gamRec.setValue({
  fieldId: 'subsidiary',
  value: '1'
});
gamRec.setText({
  fieldId: 'class',
  text: 'Class US'
});
gamRec.setText({
  fieldId: 'department',
  text: 'Department US'
});
gamRec.setText({
  fieldId: 'location',
  text: 'Location US'
});
gamRec.setValue({
  fieldId: 'destinationaccount',
  text: '6'
});
var id = gamRec.save();

// Update Record
var gamRec = record.load({
  type: record.Type.GLOBAL_ACCOUNT_MAPPING,
  id: '102'
});
gamRec.setValue({
  fieldId: 'effectivedate',
...
value: new Date('5/5/2005')
});
gamRec.setText({
    fieldId: 'sourceaccount',
    value: 'Advances Paid'
});
gamRec.setText({
    fieldId: 'destinationaccount',
    value: 'ABN Withholding'
});

var id = gamRec.save();

// Delete Record
var gamRec = record.delete({
    type: record.Type.GLOBAL_ACCOUNT_MAPPING,
    id: '102'
});

// Copy Record
var gamRec = record.copy({
    type: record.Type.GLOBAL_ACCOUNT_MAPPING,
    id: '103'
});
gamRec.setValue({
    fieldId: 'effectivedate',
    value: new Date('5/5/2005')
});
gamRec.setText({
    fieldId: 'sourceaccount',
    text: 'Advances Paid'
});
gamRec.setText({
    fieldId: 'destinationaccount',
    text: 'ABN Withholding'
});
var id = gamRec.save();

Global Inventory Relationship

Note: This topic applies to all versions of SuiteScript.

The internal ID for this record is globalinventoryrelationship.

The Global Inventory Relationship record is available only when the following features are enabled:

- Multi-Location Inventory
- Advanced Inventory Management
- Intercompany Cross-Subsidiary Fulfillment

Use the Intercompany Cross-Subsidiary Fulfillment feature in your NetSuite OneWorld account to fulfill orders and receive returns across multiple subsidiaries. Use of this feature means that orders are not limited to being fulfilled from locations within the originating sales subsidiary. Instead, you can fulfill a single sales order from locations in multiple subsidiaries.
For help working with this record in the UI, see the help topics Intercompany Cross-Subsidiary Fulfillment, Advanced Inventory Management, and Multi-Location Inventory.

See the SuiteScript Records Browser for all internal IDs associated with this record. See Supported Operations for additional details about scriptable elements for this record.

**Supported Script Types**

The Global Inventory Relationship record is scriptable in both client and server SuiteScript.

**Supported Operations**

The Global Inventory Relationship record can be created, read, edited, deleted, and searched using SuiteScript. It cannot be copied.

**Usage Notes**

The following elements of the Global Inventory Relationship record have limitations on scripting:

- Only searching is supported for scripts on the System Notes sublist.
- The originatingsubsidiary field is mandatory and can no longer be updated after it has been created.
- The inventorysubsidiary is mandatory and can no longer be updated after it has been created.
- The allowcrosssubfulfillment field is disabled and hidden by default but can be exposed on a Global Inventory Relationship form that has been customized.
- The allowcrosssubcustomerreturn is disabled and hidden by default but can be exposed on a Global Inventory Relationship form that has been customized.

**Fields**

The following fields are scriptable for this record:

- originatingsubsidiary (type: select, label: Originating Subsidiary)
- inventorysubsidiary (type: select, label: InventorySubsidiary)
- allowcrosssubfulfillment (type: checkbox, label: Allow Cross-Subsidiary Fulfillment)
- allowcrosssubcustomerreturn (type: checkbox, label: Allow Cross-Subsidiary Customer Return)
- allocationsfulfillment (type: checkbox, label: All Fulfillment Locations)
- allocationscustomerreturn (type: checkbox, label: All Customer Return Locations)
- externalid (type: text), isinactive (type: checkbox, label: Inactive)

**Subtabs**

The following subtabs are scriptable for this record:

- invtlocfulfillment (Fulfillment)
- invtloccustreturn (Customer Return)

**Sublists**

The following sublists are scriptable for this record:

- fulfillment (Location under Fulfillment tab)
customerreturn (Location under Customer Return tab)

Additional Notes

- A Global Inventory Relationship (GIR) record's originatingsubsidiary field cannot be set with the same value as its inventorysubsidiary field.
- A GIR record's originatingsubsidiary and inventorysubsidiary combination should be unique. Global Inventory Relationship records cannot be duplicated.
- A GIR record cannot have the allowcrossubfulfillment box and allowcrossubcustomerreturn box cleared at the same time. At least one should be checked. Both can be checked if applicable.
- A GIR record cannot be deleted when a sales order and/or return authorization has been created using the relationship record.
- A GIR record's allowcrossubfulfillment box can be updated only when one of the following are true:
  - No sales order uses the GIR record, or
  - All sales orders using the GIR record have been fully fulfilled and closed.
- A GIR record's allowcrossubcustomerreturn check box can be updated only when one of the following are true:
  - No return authorization uses the GIR record, or
  - All return authorizations using the GIR record have been fully received and closed.
- You cannot clear the allowcrossubfulfillment box when one or more locations are on the fulfillment sublist.
- You cannot clear the allowcrossubcustomerreturn box when one or more locations are on the customerreturn sublist.
- You cannot remove a location from the fulfillment sublist when there is an existing cross-subsidiary sales order, that is not fully fulfilled and closed, that uses the corresponding GIR with the given location.
- You cannot remove a location from the customerreturn sublist when there is an existing cross-subsidiary return authorization, that is not fully fulfilled and closed, that uses the corresponding GIR with the given location.

Code Sample

The following samples illustrate how to create a Global Inventory Relationship (GIR) record. This code creates two GIR records with the following parameters:

- id1 is a relationship between US Subsidiary as the originating subsidiary and the AU subsidiary as the inventory subsidiary. It uses default values.
- id2 is a relationship between the AU subsidiary as the originating subsidiary and US subsidiary as the inventory subsidiary. It sets specific values such as enabling the Inactive field.

```javascript
var US_SUBSIDIARY = 3;
var AU_SUBSIDIARY = 1;

var girRec = record.create({
  type: record.Type.GLOBAL_INVENTORY_RELATIONSHIP,
});
girRec.setValue({
  fieldId: 'originatingsubsidiary',
  value: US_SUBSIDIARY
});
```
girRec.setValue({
    fieldId: 'inventorysubsidiary',
    value: AU_SUBSIDIARY
});
var id1 = girRec.save();

var girRec2 = record.create({
    type: record.Type.GLOBAL_INVENTORY_RELATIONSHIP,
});
girRec2 setValue({
    fieldId: 'originatingsubsidiary',
    value: AU_SUBSIDIARY
});
girRec2 setValue({
    fieldId: 'inventorysubsidiary',
    value: US_SUBSIDIARY
});
girRec2 setValue({
    fieldId: 'externalid',
    value: 'AU_US_GIR'
});
girRec2 setValue({
    fieldId: 'isinactive',
    value: true
});
girRec2 setValue({
    fieldId: 'allocationsfulfillment',
    value: true
});
girRec2 setValue({
    fieldId: 'allocationscustomerreturn',
    value: false
});
var id2 = girRec2.save();

The following sample shows how to create a simple search that returns GIR records that are set as Inactive.

var mySearch = search.create({
    type: record.Type.GLOBAL_INVENTORY_RELATIONSHIP,
    filters: [{
        name: 'isinactive',
        operator: search.Operator.IS,
        values: true
    }]
});

Government-Issued ID Tracking

Note: This topic applies to all versions of SuiteScript.

The Government-Issued ID Tracking feature enables you to track a variety of government-issued identification information about your employees to ensure that you have accurate documentation for your employees' travel, driving qualifications, and tax compliance.
For help working with this record in the UI, see the help topic Government-Issued ID Tracking.

The following record types are related to the Government-Issued ID Tracking feature:

- Driver's License
- Passport
- Other Government-Issued ID

**Driver’s License**

**Note:** This topic applies to all versions of SuiteScript.

The driver’s license record captures the driver’s license number for a particular employee.

The internal ID for this record is `driverslicense`. For more details about this record and how to work with it in the UI, see the help topic Government-Issued ID Tracking.

This record becomes available when either the Basic Government-Issued ID Tracking or Advanced Government-Issued ID Tracking features are enabled.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is scriptable in both server and client SuiteScript. However, user event scripts are not supported.

**Supported Functions**

This record is fully scriptable, which means it can be created, edited, copied, deleted, and searched.

**Code Sample**

The following sample shows how to create a new driver's license for an employee.

```javascript
var dlRecord = record.create({
  type: 'driverslicense'
});
dlRecord.setValue({
  fieldId: 'employee',
  value: '123'
});
dlRecord.setValue({
  fieldId: 'numbervalue',
});
dlRecord.setValue({
  fieldId: 'numbervalue',
});
```
value: '123456789'
});

dlRecord.setValue({
    fieldId: 'nameondocument',
    value: 'John Doe'
});

dlRecord.setValue({
    fieldId: 'issuer',
    value: 'United States Government'
});

dlRecord.setValue({
    fieldId: 'class',
    value: 'M3'
});

dlRecord.setValue({
    fieldId: 'dateofissue',
    value: new Date('10/12/2017')
});

dlRecord.setValue({
    fieldId: 'expirationdate',
    value: new Date('10/12/2027')
});

dlRecord.setValue({
    fieldId: 'description',
    value: 'This is a Drivers License'
});

var driversLicenseInternalId = dlRecord.save();

### Passport

**Note:** This topic applies to all versions of SuiteScript.

The passport record captures the passport number for a particular employee. The internal ID for this record is **passport**. For more details about this record and how to work with it in the UI, see the help topic [Government-Issued ID Tracking](https://www.netsuite.com/). This record becomes available when either the Basic Government-Issued ID Tracking or Advanced Government-Issued ID Tracking features are enabled.

See the [SuiteScript Records Browser](https://www.netsuite.com/) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](https://www.netsuite.com/).

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](https://www.netsuite.com/)
- [SuiteScript 2.0 Custom Forms](https://www.netsuite.com/)
- [N/record Module](https://www.netsuite.com/)

### Supported Script Types

This record is scriptable in both server and client SuiteScript. However, user events are not supported.
Supported Functions

This record is fully scriptable, which means it can be created, edited, copied, deleted, and searched.

Code Sample

The following sample shows how to create a new passport number for an employee.

```javascript
var passportRec = record.create(
    {type: 'passport'});
passportRec.setValue(
    {fieldId: 'employee',
     value: '123'});
passportRec.setValue(
    {fieldId: 'numbervalue',
     value: '123456789'});
passportRec.setValue(
    {fieldId: 'issuer',
     value: 'United States Government'});
passportRec.setText(
    {fieldId: 'nationality',
     value: 'American'});
passportRec.setValue(
    {fieldId: 'nameondocument',
     value: 'John Doe'});
passportRec.setValue(
    {fieldId: 'dateofissue',
     value: new Date('10/12/2017')})
passportRec.setValue(
    {fieldId: 'expirationdate',
     value: new Date('10/12/2027')})
passportRec.setValue(
    {fieldId: 'description',
     value: 'This is a Passport'});

var passportInternalId = passportRec.save();
```

Other Government-Issued ID

**Note:** This topic applies to all versions of SuiteScript.

The other government-issued ID record captures ID types other than driver’s license and passport. The internal ID for this record is `othergovernmentissuedid`. For more details about this record and how to work with it in the UI, see the help topic Government-Issued ID Tracking.
This record becomes available when the Advanced Government-Issued ID Tracking feature is enabled.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is scriptable in both server and client SuiteScript. However, user events are not supported.

**Supported Functions**

This record is fully scriptable, which means it can be created, edited, copied, deleted, and searched.

**Code Sample**

The following sample shows how to create a new government-issued ID type.

```javascript
var otherGovIdRec = record.create({
  type: 'othergovernmentissuedid'
});
otherGovIdRec.setValue({
  fieldId: 'employee',
  value: '123'
});
otherGovIdRec.setText({
  fieldId: 'idtype',
  value: 'Social Insurance Number (SIN)'
});
otherGovIdRec.setValue({
  fieldId: 'numbervalue',
  value: '123456789'
});
otherGovIdRec.setValue({
  fieldId: 'nameondocument',
  value: 'John Doe'
});
otherGovIdRec.setValue({
  fieldId: 'issuer',
  value: 'United States Government'
});
otherGovIdRec.setValue({
  fieldId: 'expirationdate',
  value: new Date('10/12/2027')
});
```
otherGovIdRec.setValue(
    {
        fieldId: 'description',
        value: 'This is a Other Government ID'
    });
var otherGovIdInternalId = otherGovIdRecord.save();

Group

Note: This topic applies to all versions of SuiteScript.

You use the group record to define groups of contacts, customers, employees, partners, or vendors.
The internal ID for this record is entitygroup.
In the UI, you access this record at Lists > Relationships > Group > New.
For help working with this record in the UI, see the help topic Working with Groups.
See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The group record is scriptable in server SuiteScript only. None of the user events are supported.

Supported Functions

This record is fully scriptable, which means it can be created, read, updated, deleted, and searched.

Usage Notes

Be aware of the following details when working with this record:

- The grouptype field is required and must be set either to “static” or “dynamic.”
- If the grouptype is dynamic, you can never manually add to the Members sublist (because the group members are determined by a saved search). This behavior is the same as in the UI.
- If the grouptype is static, you can create lines in the Members sublist during the time that you are creating the record, but you cannot modify the sublist during updates. This behavior differs from the UI, which does allow manual updates after the record is created.
  To add members to a group in SuiteScript after it has been created, you can use code like the following:
record.attach({
    record: {
        type: record.Type.CUSTOMER,
        id: <someCustomerId>
    },
    to: {
        type: 'entitygroup',
        id: <someGroupId>
    }
});

record.detach({
    record: {
        type: record.Type.CUSTOMER,
        id: <someCustomerId>
    },
    from: {
        type: 'entitygroup',
        id: <someGroupId>
    }
});

## Code Sample

The following sample shows how to create a static group record.

```javascript
var initValues = new Array();
initValues.grouptype = 'Employee';
initValues.dynamic = 'false';
var staticRecGroup = record.create({
    type: 'entitygroup',
    defaultValues: initValues
});
staticRecGroup.setValue({
    fieldId: 'groupname',
    value: 'WC1'
});
staticRecGroup.setValue({
    fieldId: 'subsidiary',
    value: '1'
});
staticRecGroup.setValue({
    fieldId: 'ismanufacturingworkcenter',
    value: true
});
staticRecGroup.setValue({
    fieldId: 'machineresources',
    value: '11'
});
staticRecGroup.setValue({
    fieldId: 'laborresources',
    value: '22'
});
var recId = staticRecGroup.save();
```


Inbound Shipment

**Note:** This topic applies to all versions of SuiteScript.

Inbound shipments provide visibility of in-transit inventory and the status of a shipment. Items from multiple purchase orders can be assigned to an incoming shipment and bulk received and billed from within the record. Accounting support is also provided for the transfer of ownership of inbound inventory prior to receipt, through the ownership transfer record. For information about this record type, see Ownership Transfer.

The internal ID for this record is `inboundshipment`.

The inbound shipment record is available in NetSuite and NetSuite OneWorld accounts when the Inbound Shipment Management feature is enabled.

For help working with this record in the UI, see the help topic Inbound Shipment Management.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The inbound shipment record is supported in client and server SuiteScript.

**Supported Operations**

This record can be created, read, updated, deleted, and searched using SuiteScript. It cannot be copied or transformed.

**Usage Notes**

A script that adds a line item to an inbound shipment should include both the purchase order ID and the line number for the item in the purchase order items sublist.

**Field Definitions**

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

**Code Sample**

The following sample shows a typical workflow for an inbound shipment. This example does the following:

- Creates the inbound shipment record.
- Adds line items from a purchase order.
- Enters information into the primary information fields.
- Updates the status field.
- Updates the expected shipping date.
- Creates a partial ownership transfer.
- Receives the inbound shipment.

```javascript
var inboundShipment = record.create({
  type: record.Type.INBOUND_SHIPMENT,
  isDynamic: true
});
var purchaseOrder = record.load({
  type: record.Type.PURCHASE_ORDER,
  id: 1718,
  isDynamic: true
});
var itemLineCount = purchaseOrder.getLineCount({
  sublistId: 'item'
});
for (var i = 1; i <= itemLineCount; i++) {
  inboundShipment.selectNewLine({
    sublistId: 'items'
  });
  inboundShipment.setCurrentSublistValue({
    sublistId: 'items',
    fieldId: 'purchaseorder',
    value: purchaseOrder.getId()
  });
  inboundShipment.setCurrentSublistValue({
    sublistId: 'items',
    fieldId: 'shipmentitem',
    value: purchaseOrder.getSublistValue({
      sublistId: 'item',
      fieldId: 'lineuniquekey',
      line: i
    })
  });
  inboundShipment.commitLine({
    sublistId: 'items'
  });
}
var inboundShipmentId = inboundShipment.save();

var inboundShipmentUpdate = record.load({
  type: record.Type.INBOUND_SHIPMENT,
  id: inboundShipmentId,
  isDynamic: true
});
inboundShipmentUpdate.setValue({
  fieldId: 'shipmentstatus',
  value: 'inTransit'
});
inboundShipmentUpdate.setValue({
  fieldId: 'externaldocumentnumber',
  value: 'EDN645'
});
inboundShipmentUpdate.setValue({
  fieldId: 'expectedshippingdate',
  value: new Date('8/2/2017')
});
```
inboundShipmentUpdate.selectLine({
    sublistId: 'items',
    line: 1
});
inboundShipmentUpdate.setCurrentSublistValue({
    sublistId: 'items',
    fieldId: 'receivinglocation',
    value: 6
});
inboundShipmentUpdate.setCurrentSublistValue({
    sublistId: 'items',
    fieldId: 'quantityexpected',
    value: 1
});
inboundShipmentUpdate.setCurrentSublistValue({
    sublistId: 'items',
    fieldId: 'expectedrate',
    value: 10.5
});
inboundShipmentUpdate.commitLine({
    sublistId: 'items'
});
var recId = inboundShipmentUpdate.save();

var takeOwnership = record.load({
    type: record.Type.BULK_OWNERSHIP_TRANSFER,
    id: inboundShipmentId,
    isDynamic: true
});
takeOwnership.selectLine({
    sublistId: 'items',
    line: 2
});
takeOwnership.setCurrentSublistValue({
    sublistId: 'items',
    fieldId: 'process',
    value: 'F'
});
takeOwnership.commitLine({
    sublistId: 'items'
});
var recId2 = takeOwnership.save();

var bulkReceive = record.load({
    type: record.Type.RECEIVE_INBOUND_SHIPMENT,
    id: inboundShipmentId,
    isDynamic: true
});
bulkReceive.selectLine({
    sublistId: 'receiveitems',
    line: 2
});
bulkReceive.setCurrentSublistValue({
    sublistId: 'receiveitems',
    line: 2
});
Intercompany Allocation Schedule

**Note:** This topic applies to all versions of SuiteScript.

Intercompany allocation schedules transfer a balance from one source subsidiary account to multiple destination subsidiaries for costs that are shared between subsidiaries.

The internal ID for this record is `intercompallocationschedule`.

The intercompany allocation schedule record is available only in OneWorld accounts and when the Accounting Periods and Expense Allocation features are enabled. An administrator can enable these features at Setup > Company > EnableFeatures > Accounting subtab. For information about expense allocation, see the help topic Expense Allocation Overview.

For help working with this record in the UI, see the help topic Creating Intercompany Allocation Schedules.

If the Statistical Accounting feature is enabled, the intercompany allocation schedule record includes fields that enable you to base the weight for the allocation on the balance of a statistical account through statistical journals or as an absolute value. If the Dynamic Allocation feature is also enabled, the weight is dynamically calculated when the intercompany allocation journal is generated. An administrator can enable these features at Setup > Company > EnableFeatures > Accounting under Advanced Features. For information about statistical allocation schedules, see the help topic Working with Allocation Schedules Weighted by the Balance of a Statistical Account.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is scriptable in both client and server SuiteScript.

If the Dynamic Allocation feature is enabled, the currency field is not scriptable. None of the fields from the History subtab on the intercompany allocation schedule record in the UI are not scriptable.
Supported Functions

This record is partially scriptable. It can be created, read, updated, and deleted using SuiteScript. It cannot be copied, updated, searched, or transformed.

Code Samples

The following samples show how to create a statistical intercompany allocation schedule that divides the balance of one account and transfers the amounts into two other subsidiaries.

```javascript
var schedRec = record.create({
    type: record.Type.INTERCOMP_ALLOCATION_SCHEDULE,
    defaultValues: false
});
schedRec.setValue({
    fieldId: 'name',
    value: 'Allocation Schedule A'
});
schedRec.setValue({
    fieldId: 'subsidiary',
    value: '1'
});

schedRec.setSublistValue({
    sublistId: 'allocationsource',
    fieldId: 'account',
    line: 1,
    value: '24'
});

schedRec.setSublistValue({
    sublistId: 'allocationdestination',
    fieldId: 'subsidiary',
    line: 1,
    value: '6'
});

schedRec.setSublistValue({
    sublistId: 'allocationdestination',
    fieldId: 'weight',
    line: 1,
    value: '50.00'
});

schedRec.setSublistValue({
    sublistId: 'allocationdestination',
    fieldId: 'subsidiary',
    line: 1,
    value: '32'
});

schedRec.setSublistValue({
    sublistId: 'allocationdestination',
    fieldId: 'weight',
    line: 1,
    value: '50.00'
});

recordId = schedRec.save();
```
Inventory Number

**Note:** This topic applies to all versions of SuiteScript.

The term inventory number refers to a serial number or a lot number. An inventory number record uniquely identifies an item in physical inventory with a serial number, or uniquely identifies a group of items with a lot number.

For help working with this record in the UI, see the help topics *Serial Numbered Items* and *Lot Numbered Items*.

The internal ID for this record is `inventorynumber`.

See the *SuiteScript Records Browser* for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic *Working with the SuiteScript Records Browser* in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

Client SuiteScript is not supported for the inventory number record. It is scriptable in server SuiteScript only.

### Supported Functions

The inventory number record is partially scriptable. It can be updated and searched using SuiteScript. Copy, Create, and Delete are not allowed for this record.

### Item Account Mapping

**Note:** This topic applies to all versions of SuiteScript.

For accounts using Multi-Book Accounting, the item account mapping record enables you to configure secondary accounting books to post to accounts different from the primary book. These mappings are used by transactions where the item determines the account to which the transaction posts.

Using this record requires that, as part of your Multi-Book configuration, you select the Chart of Accounts Mapping option at Setup > Enable Features, on the Accounting subtab, in addition to the other setup steps required for Multi-Book Accounting.

In the UI, you access this record at Setup > Accounting > Item Account Mappings > New.

For help working with this record in the UI, see the help topic *Item Account Mapping*.

The internal ID for this record is `itemaccountmapping`.

See the *SuiteScript Records Browser* for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic *Working with the SuiteScript Records Browser* in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types

This record is scriptable in server SuiteScript only.
All three user events are supported: before Load, before Submit, and afterSubmit.

Supported Functions

The item account mapping record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Code Samples

The following samples show how to create item account mapping records and perform other basic tasks.

```javascript
// Get Record
var itemMapRec1 = record.load(
    { type: record.Type.ITEM_ACCOUNT_MAPPING,
    id: '105' });

// Create Record
var itemMapRec2 = record.create(
    { type: record.Type.ITEM_ACCOUNT_MAPPING });

itemMapRec2.setValue(
    { fieldId: 'effectivedate',
    value: new Date('4/4/2004') });

itemMapRec2.setValue(
    { fieldId: 'accountingbook',
    value: '2' });

itemMapRec2.setValue(
    { fieldId: 'sourceaccount',
    value: '118' });

itemMapRec2.setText(
    { fieldId: 'itemaccount',
    text: 'Asset' });

itemMapRec2.setValue(
    { fieldId: 'subsidiary',
    value: '3' });

itemMapRec2.setText(
    { fieldId: 'class',
    text: 'Class US' });

itemMapRec2.setText(
    { fieldId: 'department',
    text: 'Department US' });
```
itemMapRec2.setText({
  fieldId: 'location',
  text: 'Location US'
});
itemMapRec2.setValue({
  fieldId: 'destinationaccount',
  value: '118'
});
var id = itemMapRec2.save();

// Update Record
var itemMapRec3 = record.load({
  type: record.Type.ITEM_ACCOUNT_MAPPING,
  id: '201'
});
itemMapRec3.setValue({
  fieldId: 'effectivedate',
  value: new Date('5/5/2005')
});
itemMapRec3.setValue({
  fieldId: 'subsidiary',
  value: '1'
});
itemMapRec3.setText({
  fieldId: 'itemaccount',
  text: 'Discount'
});
itemMapRec3.setText({
  fieldId: 'sourceaccount',
  text: 'Advertising'
});
itemMapRec3.setText({
  fieldId: 'destinationaccount',
  text: 'Advertising'
});
var id = itemMapRec3.save();

// Copy Record
var itemMapRec4 = record.copy({
  type: record.Type.ITEM_ACCOUNT_MAPPING,
  id: '201'
});
itemMapRec4.setValue({
  fieldId: 'effectivedate',
  value: new Date('6/6/2005')
});
itemMapRec4.setText({
  fieldId: 'itemaccount',
  text: 'Asset'
});
itemMapRec4.setValue({
  fieldId: 'sourceaccount',
  value: '118'
});
Item Collection

**Note:** This topic applies to all versions of SuiteScript.

Item collections are a general purpose way of grouping catalog items. They are particularly useful in setting up Personalized Catalog Views.

For help working with this record in the user interface, see the help topic [Item Collections](#).

In the UI, you can go to Lists > Accounting > Item Collections to access this record.

The internal ID for this record is `itemcollection`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

**Supported Script Types**

The item collection record is scriptable in both client and server SuiteScript.

**Supported Functions**

The following SuiteScript functionality is supported:

- Read
- Create
- Update
- Delete
- Search
Usage Notes

You must enable the Inventory feature on the Items & Inventory subtab at Setup > Company > Enable Features to be able to script with this record. This record cannot be used to add items to an item collection. Adding items is done using the Item Collection Item Map record.

The Item Collection record has two sublists:

- **Items** – this sublist cannot be accessed directly using SuiteScript. However, it can be accessed using the separate Item Collection Item Map record.
- **Customer Segments** – this sublist cannot be accessed directly using SuiteScript.

Script Sample

The following code snippets create an item collection record with the name Item Collection 1.

```javascript
// SS 1.0
var collection = nlapiLoadRecord("itemcollection", 1);
collection.setFieldValue("name", "Item Collection 1");
nlapiSubmitRecord(collection);

// SS 2.0
require(['N/record'], function(record){
    var collection = record.load({type: "itemcollection", id: "1"});
    collection.setValue({fieldId: "name", value: "ItemCollection 1"});
    collection.save();
});
```

Item Collection Item Map

**Note:** This topic applies to all versions of SuiteScript.

This record holds the relationship between an item collection and the items it contains. It is not accessible from the UI.

The internal ID for this record is `itemcollectionitemmap`.

See the [SuiteScript Records Browser](https://www.netsuite.com) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](https://www.netsuite.com) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- [SuiteScript 2.0 Scripting Records and Subrecords](https://www.netsuite.com)
- [SuiteScript 2.0 Custom Forms](https://www.netsuite.com)
- [N/record Module](https://www.netsuite.com)

Supported Script Types

The item collection record is scriptable in both client and server SuiteScript.
Supported Functions

The following SuiteScript functionality is supported:

- Read
- Create
- Delete
- Search

Usage Notes

You must enable the Inventory feature on the Items & Inventory subtab at Setup > Company > Enable Features to be able to script with this record.

Script Sample

The following code snippets deletes the item collection item map record that has the ID of 1.

```javascript
// SS 1.0
var mapping = nlapiLoadRecord("itemcollectionitemmap", 1);
nlapiDeleteRecord("itemcollectionitemmap", 1);

// SS 2.0
require(["N/record"], function(record){
    var mapping = record.load({type: "itemcollectionitemmap", id: 1});
    record.delete({type: "itemcollectionitemmap", id: 1});
});
```

Item Location Configuration

Note: This topic applies to all versions of SuiteScript.

The internal ID for this record is `itemlocationconfiguration`.

The item location configuration record is available only when the following features are enabled:

- Multi-Location Inventory
- Advanced Inventory Management
- Advanced Item Location Configuration

The Advanced Item Location Configuration feature supports tracking of location attributes data for items in accounts where the Multi-Location Inventory and Advanced Inventory Management features are enabled. For information about these features, see the help topics Advanced Inventory Management and Multi-Location Inventory.

For help working with this record in the UI, see the following help topics:

- Advanced Item Location Configuration
- Enabling Advanced Item Location Configuration
- Using Advanced Item Location Configuration
- Working with Item Location Attributes
- Duplicate Item Attributes from a Single Location
- Duplicate Item Attributes from Multiple Locations
- Edit Item Location Attributes

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The item location configuration record is scriptable in both client and server SuiteScript.

**Supported Operations**

The item location configuration record can be created, read, edited, deleted, and searched using SuiteScript. It cannot be copied or transformed.

**Code Sample**

The following samples illustrate how to make changes on an item location configuration record.

```javascript
// OW account, subsidiary (id=1) "Parent Company", locations: (id=6) "US ONLY Location"
var itemLocConfigRec = record.create({
    type: record.Type.ITEM_LOCATION_CONFIGURATION,
    defaultValues: {
        item: '18'
    }
});
itemLocConfigRec.setValue({
    fieldId: 'subsidiary',
    value: '1'
});
itemLocConfigRec.setValue({
    fieldId: 'location',
    value: '6'
});
itemLocConfigRec.setValue({
    fieldId: 'itemid',
    value: '18'
});
itemLocConfigRec.setValue({
    fieldId: 'name',
    value: 'ilc record1'
});
var recId = itemLocConfigRec.save();

// Update Record
var itemLocConfigRec2 = record.load({
    id: recId
});
```

---

Lists | 326
Item Revision

Note: This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Bill of Materials Member Control for Assembly Items.

The internal ID for this record is `itemrevision`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

Client SuiteScript is not supported for this record. It is scriptable in server SuiteScript only.

Supported Functions

The item revision record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.
Code Sample

The following sample shows how to create an item revision record:

```javascript
function afterSubmit(type) {
    var itemRev = record.create({
        type: record.Type.ITEM_REVISION
    });
    itemRev.setValue({
        fieldId: 'name',
        value: 'revision name 222'
    });
    itemRev.setValue({
        fieldId: 'item',
        value: '109'
    });
    itemRev.setValue({
        fieldId: 'memo',
        value: 'revision memo'
    });
    itemRev.setValue({
        fieldId: 'effectivedate',
        value: new Date('3/4/2012')
    });
    var id = itemRev.save();
}
```

Kudos Feature Records

**Note:** This topic applies to all versions of SuiteScript.

The Kudos feature enables your employees to give their colleagues social-based recognition for projects completed or a job well done.

For help working with this record in the UI, see the help topic Kudos.

The following record types are related to the Kudos feature:

- Kudos
- Organization Value

Kudos

**Note:** This topic applies to all versions of SuiteScript.

A Kudos is an act of recognition from one employee to another.

The internal ID for this record is kudos. For more details about this record and how to work with it in the UI, see the help topic Kudos.

This record becomes available when the Kudos feature is enabled.

See the SuiteScript Records Browser for all internal IDs associated with this record.
Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

## Supported Script Types

This record is scriptable in server SuiteScript only.

## Supported Functions

This record can be read, created, edited, copied, deleted, and searched using SuiteScript. Note that only the inactive field can be edited.

## Organization Value

Note: This topic applies to all versions of SuiteScript.

An organization value is a value that you associate with a Kudos when you give a Kudos to a colleague.

The internal ID for this record is `organizationvalue`. For more details about this record and how to work with it in the UI, see the help topic Kudos.

This record becomes available when the Kudos feature is enabled.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

## Supported Script Types

This record is scriptable in server SuiteScript only.

## Supported Functions

This record can be read, created, edited, copied, deleted, and searched using SuiteScript.
Location

Note: This topic applies to all versions of SuiteScript.

The Locations feature enables you to track information about employees and transactions for multiple offices or warehouses. For example, you can create locations for the corporate office and all of the sales offices, or for warehouses in several states. You can associate transactions and employees with each location, and then filter report data by location.

For help working with this record in the UI, see the help topic Locations Overview.

The internal ID for this record is location.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

Client SuiteScript is not supported for this record. It is scriptable in server SuiteScript only.

Supported Functions

The location record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Manufacturing Cost Template

Note: This topic applies to all versions of SuiteScript.

To work with this record, the Manufacturing Routing and Work Center feature must be enabled at Setup > Company > Enable Features, on the Items & Inventory tab.

In the UI, this record is accessed by going to Lists > Supply Chain > Manufacturing Cost Template.

For help working with this record in the UI, see the help topic Creating Manufacturing Cost Templates.

The internal ID for this record is manufacturingcosttemplate.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
Lists

■ SuiteScript 2.0 Custom Forms
■ N/record Module

Supported Script Types
The manufacturing cost template record is scriptable in server SuiteScript only.

Supported Functions
The manufacturing cost template record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Manufacturing Routing

Note: This topic applies to all versions of SuiteScript.

To work with this record, the Manufacturing Routing and Work Center feature must be enabled at Setup > Company > Enable Features, on the Items & Inventory tab.

In the UI, this record is accessed by going to Lists > Supply Chain > Manufacturing Routing.

For help working with this record in the UI, see the help topic Manufacturing Routing.

The internal ID for this record is manufacturingrouting.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
■ SuiteScript 2.0 Scripting Records and Subrecords
■ SuiteScript 2.0 Custom Forms
■ N/record Module

Supported Script Types
The manufacturing routing record is scriptable in server SuiteScript only.

Supported Functions
The manufacturing routing record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Merchandise Hierarchy Level

Note: This topic applies to all versions of SuiteScript.

You can use the Merchandise Hierarchy feature to help you plan, price, report on, and analyze your product inventory by categorizing inventory items in your enterprise.
A merchandise hierarchy level provides a useful way to categorize hierarchy nodes and determine the relationships between inventory items.

The Merchandise Hierarchy Level record is available only when the following features are enabled:

- Merchandise Hierarchy
- Custom Segments

For help working with this record in the UI, see the help topic Merchandise Hierarchy.

The internal ID for this record is `merchandisehierarchylevel`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

### Supported Script Types

Client SuiteScript is not supported for this record. It is scriptable in server SuiteScript only.

### Supported Functions

The Merchandise Hierarchy Level record can be read, created, updated, deleted, and searched using SuiteScript.

### Code Sample

The following sample shows how to create a new Merchandise Hierarchy Level record:

```javascript
require(['N/record'], function(record){
    var level = record.create({
        type: record.Type.MERCHANDISE_HIERARCHY_LEVEL,
        isDynamic: true
    });
    level.setValue({
        fieldId: 'name',
        value: 'Level 1'
    });
    level.setValue({
        fieldId: 'description',
        value: 'Description of Level 1'
    });
    level.save();
});
```

### Merchandise Hierarchy Node

**Note:** This topic applies to all versions of SuiteScript.

You can use the Merchandise Hierarchy feature to help you plan, price, report on, and analyze your product inventory by categorizing inventory items in your enterprise.
A merchandise hierarchy node provides groupings for items that share similar characteristics. They can either be a point in the hierarchy from which other nodes can be created, or the point in the hierarchy where products are assigned to.

A Merchandise Hierarchy Node record contains details of the position of the node in the hierarchy, and the version or versions in which it is included.

The Merchandise Hierarchy Node record is available only when the following features are enabled:

- Merchandise Hierarchy
- Custom Segments

For help working with this record in the UI, see the help topic Merchandise Hierarchy.

The internal ID for this record is `merchandisehierarchynode`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

**Supported Script Types**

Client SuiteScript is not supported for this record. It is scriptable in server SuiteScript only.

**Supported Functions**

The Merchandise Hierarchy Node record can be read, created, updated, deleted, and searched using SuiteScript.

**Subtabs**

The following subtabs are scriptable for this record:

- hierarchyversions (Merchandise Hierarchy Versions)

**Code Sample**

The following sample shows how to create a new Merchandise Hierarchy Node record:

```javascript
require(["N/record"], function(record) {
  var node = record.create({
    type: record.Type.MERCHANDISE_HIERARCHY_NODE,
    isDynamic: true
  });
  node.setValue({
    fieldId: 'name',
    value: 'Node 1'
  });
  node.setValue({
    fieldId: 'description',
    value: 'Description of Node 1'
  });
});
```
Merchandise Hierarchy Version

Note: This topic applies to all versions of SuiteScript.

You can use the Merchandise Hierarchy feature to help you plan, price, report on, and analyze your product inventory by categorizing inventory items in your enterprise.

A Merchandise Hierarchy Version record contains all the levels included in a hierarchy definition for a specific time period.

The Merchandise Hierarchy Version record is available only when the following features are enabled:

- Merchandise Hierarchy
- Custom Segments

For help working with this record in the UI, see the help topic Merchandise Hierarchy.

The internal ID for this record is `merchandisehierarchyversion`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

Client SuiteScript is not supported for this record. It is scriptable in server SuiteScript only.

**Supported Functions**

The Merchandise Hierarchy Version record can be read, created, updated, deleted, and searched using SuiteScript.

**Subtabs**

The following subtabs are scriptable for this record:

- hierarchylevels (Merchandise Hierarchy Levels)

**Usage Notes**

In order to assign levels to a merchandise hierarchy version, you must first create the levels.
## Code Sample

The following sample shows how to create a new Merchandise Hierarchy Version record:

```javascript
require(['N/record'], function(record){
  var version = record.create({
    type: record.Type.MERCHANDISE_HIERARCHY_VERSION,
    isDynamic: true
  });
  version.setValue({
    fieldId: 'name',
    value: 'Version 1'
  });
  version.setValue({
    fieldId: 'description',
    value: 'Description of Version 1'
  });
  version.setValue({
    fieldId: 'startdate',
    value: new Date('1/1/2030')
  });
  version.selectLine({
    sublistId:'hierarchylevels',
    line: 0
  });
  version.setCurrentSublistValue({
    sublistId: 'hierarchylevels',
    fieldId: 'isincluded',
    value: true
  });
  version.commitLine({
    sublistId: 'hierarchylevels'
  });
  version.save();
});
```

## Nexus

**Note:** This topic applies to all versions of SuiteScript.

A nexus is a tax jurisdiction where you have a business or taxable presence in. In NetSuite, it is usually defined at the country level. This type of entity is available when the Advanced Taxes feature is enabled, to allow users to manage and calculate taxes for different jurisdictions within the same NetSuite account.

For help working with this record in the UI, see the help topic Advanced Taxes.

The internal ID for this record is `nexus`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Supported Script Types

The nexus record is scriptable in server SuiteScript only.

Supported Functions

The nexus record is partially scriptable. It can be created, updated, deleted, and searched using SuiteScript. It cannot be copied.

Note Type

Note: This topic applies to all versions of SuiteScript.

This type defines a list of values that are used by the note record to set the type of note. In the UI, this is a user defined list at Setup > Sales > Setup Tasks > CRM Lists > New > Note Type.

The internal ID for this record is notetype.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The note type record is scriptable in both client and server SuiteScript.

Supported Functions

The note type record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Other Name Category

Note: This topic applies to all versions of SuiteScript.

Other name category values are used on other name records to categorize them. The list of other name records is a collection of records for people or companies who are not vendors, customers, or employees.
You can use other name records to track other people or companies to whom you write checks or from whom you receive deposits.

For example, your company might donate money to a favorite charity, so you can create an other name record for the charity. You might also list your owners and partners here if they contribute or withdraw equity.

For help working with this record in the UI, see the help topic Other Name Records.

The internal ID for this record is `othernamecategory`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The other name category record is scriptable in both client and server SuiteScript.

**Supported Functions**

The other name category record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

**Ownership Transfer**

**Note:** This topic applies to all versions of SuiteScript.

You can use the ownership transfer record to transfer ownership of an inbound shipment before it has been physically received by a receiving location. This transfer makes it possible to take ownership of an inbound shipment as soon as it has been marked as in-transit. Ownership transfers can be applied to individual items as well as entire shipments. A posting transaction, Ownership Transaction, reflects the impact of the transfer on the general ledger, summarized as: DR External In Transit Inventory account; CR Inventory Received Not Billed account.

For help working with this record in the UI, see the help topic Inbound Shipment Management.

The internal IDs for the two different ownership transfer records are:

- `bulkownershiptransfer` (the posting transaction bulk creation page)
- `ownershiptransfer` (the resulting posting transaction)

The ownership transfer record is available in NetSuite and NetSuite OneWorld accounts when the Inbound Shipment Management feature is enabled.

For information about scripting with this record in SuiteScript, see the following help topics:
### Supported Script Types

The ownership transfer record is supported in client and server SuiteScript.

### Supported Operations

The `ownershiptransfer` record can be created, deleted and searched. This record type cannot be edited in SuiteScript.

### Field Definitions

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

### Code Sample

The following samples illustrate how to create and delete an ownership transfer.

```javascript
var takeOwnership = record.create({
    type: record.Type.BULK_OWNERSHIP_TRANSFER,
    defaultValues: 267  // set id of inbound shipment you want transferred
});
takeOwnership.setSublistValue({
    sublistId: 'process',
    fieldId: 'receivingLocation',
    line: 2,
    value: false         // second line won't be transferred
});
var recId = takeOwnership.save();

record.delete({
    type: record.Type.BULK_OWNERSHIP_TRANSFER,
    id: 720
});
```

### Partner Category

**Note:** This topic applies to all versions of SuiteScript.

To create new partner categories, go to Setup > Sales > Setup Tasks > CRM Lists (Administrator).

The note type record is scriptable in both client and server SuiteScript.
For help working with this record in the UI, see the help topic Creating a Partner Record.

The internal ID for this record is partnercategory.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The partner category record is scriptable in both client and server SuiteScript.

### Supported Functions

The partner category record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

### Payment Instruments

**Note:** This topic applies to all versions of SuiteScript.

The Payment Instruments feature makes using various payments methods more straightforward and, in the case of payment cards, more secure. Payment instrument is a set of fields that can be used to charge a shopper without any action on the shopper’s side. For more details and instructions on how to enable the feature, see the help topic Payment Instruments.

For help working with this record in the UI, see the help topic Payment Instruments.

The following Payment Instruments records are supported in SuiteScript:

- **General Token** - the internal ID is generaltoken
- **Payment Card** - the internal ID is paymentcard
- **Payment Card Token** - the internal ID is paymentcardtoken

### Supported Script Types

The Payment Instruments records are scriptable in both client and server SuiteScript.

### Usage Note

You cannot retrieve the value of the Payment Card Number (cardnumber) field or the Token (token) field from the original form after you save the payment instrument. For security reasons, the values of
those fields are masked. The Payment Card Number field is located on the Payment Card record, and the Token field is located on the General Token and Payment Card token records.

**Supported Functions**

The General Token, Payment Card Token, and Payment Card records can be created, read, updated, deleted, and searched by using SuiteScript. The records cannot be copied or transformed.

**Code Sample**

The following script sample shows how to create a payment instrument of the Payment Card type. The script sets the compulsory fields, and also enables Dynamic mode, so other compulsory fields are set automatically.

The script sample sets the **Preserve on File** field value to True. As a consequence, the payment instrument information is saved in the system and can be used in other transactions.

```javascript
require(['N/record'], function(record){
  function createPaymentCard(){
    var recordObj = record.create({
      type: record.Type.PAYMENT_CARD,
      isDynamic: true
    });
    recordObj.setValue({
      fieldId: 'entity',
      value: '6'
    });
    recordObj.setValue({
      fieldId: 'cardnumber',
      value: '4111111111111111'
    });
    recordObj.setValue({
      fieldId: 'expirationdate',
      value: new Date(2023, 11)
    });
    recordObj.setValue({
      fieldId: 'paymentmethod',
      value: '5'
    });
    recordObj.setValue({
      fieldId: 'nameoncard',
      value: 'John Wolfe'
    });
    recordObj.setValue({
      fieldId: 'preserveonfile',
      value: true
    });
    var recordId = recordObj.save({
      ignoreMandatoryFields: false
    });
  }
  createPaymentCard();
});
```
Payment Method

**Note:** This topic applies to all versions of SuiteScript.

You can create payment methods to add selections to the Payment Method field on transactions and in your Web store.

For help working with this record in the UI, see the help topic [Creating a Payment Method](#).

The internal ID for this record is **paymentmethod**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

**Supported Script Types**

The payment method record is scriptable in both client and server SuiteScript.

**Supported Functions**

The payment method record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Payroll Item

**Note:** This topic applies to all versions of SuiteScript.

Payroll items store values for payroll transaction line items. Payroll items are used with payroll transactions generated by the Payroll feature, but these transactions are managed by NetSuite and are generally not used in integrations.

For help working with this record in the UI, see the help topic [Using the Paycheck Journal Feature](#).

The internal ID for this record is **payrollitem**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
List

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The payroll item record is scriptable in both client and server SuiteScript.

**Supported Functions**

The payroll item record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

**Usage Notes**

The exposure of this record to SuiteScript is intended to enable global payroll solutions. You can use it along with the Paycheck Journal record to create custom payroll solutions and to support integrations with external payroll systems. For details, see the help topic Using the Paycheck Journal Feature.

**Code Samples**

The following samples create Deduction type payroll items.

```javascript
function createPayrollItemDeductionMinimal(){
    var payrollItem = record.create({
        type: record.Type.PAYROLL_ITEM
    });
    payrollItem.setValue({
        fieldId: 'subsidiary',
        value: 1
    });
    payrollItem.setValue({
        fieldId: 'itemtype',
        value: 16
    });
    payrollItem.setValue({
        fieldId: 'liabilityaccount',
        value: 27
    });
    payrollItem.setValue({
        fieldId: 'name',
        value: 'SSSitem-Deduction-Minimal'
    });
    payrollItem.setValue({
        fieldId: 'custrecord_payroll_item',
        value: 'Cust_field'
    });
    payrollItem.setValue({
        fieldId: 'externalid',
        value: 'testingexternalID'
    });
}
```
The following samples create Earning:Addition type payroll items.

```javascript
function createPayrollItemAdditionMinimal()
{
    var payrollItem = record.create({
        type: record.Type.PAYROLL_ITEM
    });
    payrollItem.setValue({
        fieldId: 'subsidiary',
        value: 1
    });
    payrollItem.setValue({
        fieldId: 'itemtype',
        value: 6
    });
    var recId = payrollItem.save();
}
```
The following sample shows how to update a payroll item.

```javascript
function updatePayrollItem()
{
    var payrollItem = record.create({
        type: record.Type.PAYROLL_ITEM,
        defaultValues: 110
    });
    payrollItem.setValue({
        fieldId: 'expenseaccount',
        value: 114
    });
    var recId = payrollItem.save();
}
```
The following sample shows how to delete a payroll item.

```javascript
function deletePayrollItem(){
    var savedSearchInternalId = 2;
    var searchresults = search.load({
        type: search.Type.PAYROLL_ITEM,
        id: savedSearchInternalId
    });
    for (var i = 0; searchresults != null && i < searchresults.length; i++) {
        var searchresult = searchresults[i];
        if (0 < searchresult.getId()){
            record.delete({
                type: searchresult.getRecordType(),
                id: searchresult.getId()
            });
        }
    }
}
```

Price Book

**Note:** This topic applies to all versions of SuiteScript.

The price book record, along with the price plan record, is used to build pricing information for subscriptions.

The price book record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features (Administrator), on the Transactions subtab.

For help working with this record in the UI, see the help topic Creating Price Books.

The internal ID for this record is `pricebook`.

See the SuiteScript Records Browser for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.
Lists

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The price book record is scriptable in server and client SuiteScript.

Supported Functions

The price book record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Price Level

Note: This topic applies to all versions of SuiteScript.

Price level defines a list of values that are used by the opportunity and item records to set the price level for a specific item. Items can be assigned different price levels — such as Employee Price or Corporate Discount Price.

When the Multiple Pricing feature is enabled, you can create up to 1000 price levels for items. Setting up multiple price levels allows greater flexibility to set different pricing for different customers.

For help working with this record in the UI, see the help topics Creating Price Levels and Setting Up Items for Multiple Price Levels.

The internal ID for this record is \texttt{pricelevel}. Search is not available on this record type.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The price level record is scriptable in both client and server SuiteScript.
Supported Functions

The price level record is partially scriptable. It can be created, updated, copied, and deleted using SuiteScript. It cannot be searched.

Price Plan

**Note:** This topic applies to all versions of SuiteScript.

The price plan record, along with the price book record, is used to build pricing information for subscriptions.

The price plan record is available only when the SuiteBilling feature is enabled at Setup > Company > Enable Features (Administrator), on the Transactions subtab.

For help working with this record in the UI, see the help topic Creating Price Plans.

The internal ID for this record is `priceplan`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record. For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The price plan record is scriptable in server and client SuiteScript.

Supported Functions

The price plan record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Pricing Group

**Note:** This topic applies to all versions of SuiteScript.

Price groups allow you to assign customer-specific price levels for groups of items. For example, you could create a pricing group called Laptops and associate the pricing group with all of your laptop items.

For help working with this record in the UI, see the help topic Pricing Groups.

The internal ID for this record is `pricinggroup`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.
Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The pricing group record is scriptable in both client and server SuiteScript.

**Supported Functions**

The pricing group record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

**Project Charge Rule**

Note: This topic applies to all versions of SuiteScript.

The project charge rule record is available only when the Project Management and Charge-Based Billing features are enabled. In the UI, you access this record on the Financial subtab of charge-based billing projects.

The internal ID for this record is `chargerule`.

Project charge rules determine the billing rate, the timing of project charges, and the stage of a charge when it is generated.

For help working with this record in the UI, see the help topic Understanding Charge Rules.

See `Project Charge Rule Record Macros` for macros associated with this record. For more information about actions and macros, see the help topic Overview of Record Action and Macro APIs.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The project charge rule record is supported in client and server SuiteScript.

**Supported Operations**

This record can be created, read, updated, deleted, copied, and searched using SuiteScript. It cannot be transformed.
Usage Notes

If charges generated from a project charge rule have been billed, the project charge rule record is read-only. The `chargeruletype` field is mandatory.

Field Definitions

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Code Samples

The following samples show how to create a time-based project charge rule.

```javascript
var chargeRule = record.create({
  type: record.Type.CHARGE_RULE,
  isDynamic: true,
  defaultValues: {
    'chargeruletype': 'TIMEBASED'
  }
});
chargeRule.setValue({
  fieldId: 'name',
  value: 'ChargeRuleName'
});
chargeRule.setValue({
  fieldId: 'project',
  value: '96'
});
var id = chargeRule.save();
```

Project Charge Rule Record Macros

The charge rule record type currently supports the `copyResources` macro.

For help working with this record in the UI, see the help topic Understanding Charge Rules.

For information about SuiteScript 2.0 record macros, see the following help topics:

- Overview of Record Action and Macro APIs
- N/record Module Members
- Macro Object Members

copyResources

| Corresponding UI Button | Copy Resources from Tasks |
Macro Description

As part of a time—based charge rule defined on a project, copies to the charge rule all project resources that have a project task assignment. This includes resources that have actual time for the project or for one of its project tasks.

If the Resource Allocations feature is enabled, this also includes resources that have a resource allocation for a project task.

If the Allow Allocated Resources to Enter Time to All Tasks project preference is enabled, this also includes resources that have a resource allocation for the project.

Returns

{notifications:[], response: {}}

Supported Script Types

Client and server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

Since

2018.1

Parameters

See the help topic Record.executeMacro(options) for details about parameters required for the execution of any macro. The copyResources macro does not support any additional parameters.

Charge Rule copyResources Macro Syntax

```javascript
// On a current record (when a time-based charge rule is opened in a browser)
require(['N/currentRecord'], function(current) {
    var rec = current.get();
    rec.executeMacro(
        {id: 'copyResources'}
    );
});

// Creating new time-based charge rule on an existing project, and executing the macro on it
require(['N/record'], function(record) {
    var chargeRule = record.create({
        type: record.Type.CHARGE_RULE,
        isDynamic: true,
        defaultValues: {
            'chargeruletype': 'TIMEBASED',
            'project': '968'
        }
    });
    chargeRule.setValue({
        fieldId: 'name',
        value: 'my time-based charge rule'
    });
    chargeRule.setValue({
        fieldId: 'ratesourcetype',
        value: 'RESOURCES'
    });
    chargeRule.executeMacro(
        {id: 'copyResources'}
    );
    chargeRule.save();
});
```
Project Expense Type

Note: This topic applies to all versions of SuiteScript.

Important: The project expense type record is part of the Job Costing and Project Budgeting feature. For information on the availability of this feature, please contact your account representative.

This record is available when the Job Costing and Project Budgeting feature is enabled at Setup > Company > Enable Features, on the Company tab. If you do not see this option in the UI, contact your account representative for assistance.

When the feature is enabled, you can access the project expense type record in the UI by choosing Setup > Accounting > Setup Tasks > Project Expense Types > New.

For help working with this record in the UI, see the help topic Creating Project Expense Types.

The project expense type record lets you create expense classifications that you can apply to projects. The internal ID for this record is `projectexpensetype`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The project expense type record is not scriptable in client SuiteScript. It is scriptable in server SuiteScript only.

Supported Functions

The project expense type record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Usage Notes

The Language sublist is available only when the Multi-Language feature is enabled at Setup > Company > Enable Features, on the Company tab. The sublist's records can be updated, but you cannot add records to the sublist through SuiteScript.

Project Revenue Rule

Note: This topic applies to all versions of SuiteScript.

The project revenue rule record is available only when the Project Management, Charge-Based Billing, and Advanced Revenue Management features are enabled. In the UI, you access this record on the Financial subtab of charge-based billing projects. There are three types of project revenue rules.
The internal IDs for these records are `laborbasedprojectrevenuerule`, `fixedamountprojectrevenuerule`, `pctcompleteprojectrevenuerule`.

Project revenue rules rely on charge rules to determine the amount of revenue to be distributed. The type of project revenue rule you create is determined by how you want the revenue to be recognized. A labor based project revenue rule recognizes the revenue created from time tracked on your project. A fixed amount project revenue rule can recognize revenue based on a date, task, or milestone. A percent complete project revenue rule recognizes revenue at a rate equivalent to the project's progress. Expense-based charge rules are not available to use with Project Revenue Recognition.

For help working with this record in the UI, see the help topics Charge-Based Project Billing and Advanced Revenue Management.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The project revenue rule record is supported in client and server SuiteScript.

**Supported Functions**

The project revenue rule record is partially scriptable. It can be created, read, updated, deleted, and searched using SuiteScript. It cannot be copied or transformed.

**Usage Notes**

For labor based project revenue rules, the filters sublist is not exposed to SuiteScript. For fixed amount project revenue rules, the recurrence field is not exposed.

When you create a new project revenue rule you must set the Project (company) field to the project/job ID. Project revenue rules are not standalone records, and therefore must be associated with a specific project.

**Code Samples**

The following sample shows how to create a project revenue rule.

```javascript
var projRevRule = record.create({
    type: 'pctcompleteprojectrevenuerule'
});
projRevRule.setValue(
    {sublistId: 'name',
     value: 'Percent Complete Rule'
    });
projRevRule.setValue(
    {sublistId: 'project',
     value: '25'
    });
```
The project work breakdown structure is available only when the Job Costing and Project Budgeting, Advanced Project Accounting, Custom Records, Custom Segments, and Activity Codes features are enabled. In the UI, you access this record at Lists > Relationships > Projects.

The internal ID for this record is `wbs`.

A work breakdown structure (WBS) provides a new subtab for project records. The WBS separates work into mutually exclusive parts. You can choose a global or monthly timeline type. Global timeline is a default option. The monthly timeline lets you to select a start and end date of the WBS. You can change the definition of the WBS lines. The monthly timeline type does not show the actuals outside of the defined time range. Unmatched revenue actuals show entries from revenue transactions with a global impact, which do not match any line of the WBS. The monthly timeline type does not display the actuals outside of the defined time range.

While selecting a task on the WBS line, add the actuals to the given line. If matching actuals by the task field fails, use the activity code. Add only one task or activity code to each line, or the system can not determine which line to use.

For help working with this record in the UI, see Project Work Breakdown Structure and Budget vs. Actual Report.

See the SuiteScript Records Browser for all internal IDs associated with this record.

See Project Work Breakdown Structure for the macros associated with this record. For more information about actions and macros, see the help topic Overview of Record Action and Macro APIs.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

See the SuiteScript Records Browser for all internal IDs associated with this record.

For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

**Supported Script Types**

The project work breakdown structure event record is supported in client and server SuiteScript.
Supported Functions

The project work breakdown structure record is partially scriptable. It can be created, read, updated, deleted, and searched using SuiteScript.

Usage Notes

Prefer using SuiteScript 2.0 before 1.0 in favor of macro capabilities. You do not have to fill the Project field in UI. Set a value to the project field when scripting, because there is no context set.

Code Samples

The following sample shows how to create a work breakdown structure record, how to create a line and how to use macros for ETC and EAC.

```javascript
require(['N/record'], function(record) {
  function createAndAssertWbs() {
    var wbs = record.create({
      type: record.Type.WBS,
      isDynamic: true
    });

    wbs.setValue({
      fieldId: 'project',
      value: '43'
    });

    wbs.setValue({
      fieldId: 'timeline',
      value: 'Global'
    });

    wbs.selectNewLine({
      sublistId: 'lines'
    });

    wbs.setCurrentSublistValue({
      sublistId: 'lines',
      fieldId: 'name',
      value: 'Line 1'
    });

    wbs.commitLine({
      sublistId: 'lines'
    });

    wbs.executeMacro({
      id: 'setAmountFieldValue',
      params: { lineIndex: 0,
                fieldId: 'cost_etc',
                value: 123.45
              }
    });
  }

  createAndAssertWbs();
});
```
var wbsId = wbs.save();

console.log(wbsId);
}

createAndAssertWbs();
);

require(['N/record'], function(record) {
  record.delete({
    type: record.Type.WBS,
    id: '18'
  });
});

Project Work Breakdown Structure Macros

**Note:** This topic applies to all versions of SuiteScript.

The project work breakdown structure record type currently supports the getAmountsFields, setAmountFieldValue, getAmountFieldValue, and getUnmatchedActuals macros.

For help working with this record in the UI, see Project Work Breakdown Structure and Budget vs. Actual Report.

For information about SuiteScript 2.0 record macros, see the following help topics:
- Overview of Record Action and Macro APIs
- N/record Module Members
- Macro Object Members

getAmountsFields

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get Amounts Fields</td>
<td>Gathers the contents of all amounts field names.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server-side scripts</th>
</tr>
</thead>
</table>

For additional information, see the help topic SuiteScript 2.0 Script Types.

Parameters

The getAmountsFields macro does not require any additional parameter.

Work Breakdown Structure getAmountsFields Macro Syntax

The following code sample returns objects:
The following code sample returns all field IDs which can be used in other macros:

```javascript
require(['N/currentRecord'], function (currentRecord){
    var record = currentRecord.get();

    var amountsFields = record.executeMacro({
        id: 'getAmountsFields'
    });
});
```

### setAmountFieldValue

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Set Amount Field Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro Description</td>
<td>Sets amount field.</td>
</tr>
<tr>
<td>Returns</td>
<td>No object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Since** 2020.1

#### Parameters

See the help topic `Record.executeMacro(options)` for details about parameters required for the execution of any macro. The `setAmountFieldValue` macro also requires the following additional parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineIndex</td>
<td>number</td>
<td>required</td>
<td>Line's index (indexed from 0)</td>
<td>2020.1</td>
</tr>
<tr>
<td>fieldId</td>
<td>string</td>
<td>required</td>
<td>field's id (obtained from the getAmountFields macro, only etc field is supported)</td>
<td>2020.1</td>
</tr>
<tr>
<td>value</td>
<td>number</td>
<td>required</td>
<td>Value</td>
<td>2020.1</td>
</tr>
</tbody>
</table>

#### Errors

See the following table for details about the `setAmountFieldValue` macro errors:

<table>
<thead>
<tr>
<th>Error</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>Invalid value type</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Required argument is missing or undefined</td>
</tr>
</tbody>
</table>
Lists

SSS_COLUMN_DOES_NOT_EXIST  Invalid fieldId
YOU_HAVE_ATTEMPTED_AN_UNSUPPORTED_ACTION  Trying set value for unsupported fields (EAC, ACTUAL)
SSS_INVALID_SUBLIST_OPERATION  Invalid lineIndex argument

Work Breakdown Structure setAmountFieldValue Macro Syntax

The following code sample updates ETC amount for the first line:

```javascript
require(['N/currentRecord'], function (currentRecord){
    var record = currentRecord.get();
    record.executeMacro({
        id: 'setAmountFieldValue',
        params: {
            lineIndex: 0,
            fieldId: '01_2020_cost_etc',
            value: 100
        }
    });
});
```

getAmountFieldValue

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Get Amount Field Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro Description</td>
<td>Gets amount field.</td>
</tr>
<tr>
<td>Returns</td>
<td>Objects</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Since</td>
<td>2020.1</td>
</tr>
</tbody>
</table>

Parameters

See the help topic Record.executeMacro(options) for details about parameters required for the execution of any macro. The getAmountFieldValue macro also requires the following additional parameters:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>lineIndex</td>
<td>number</td>
<td>required</td>
<td>Line's index (indexed from 0)</td>
<td>2020.1</td>
</tr>
<tr>
<td>fieldId</td>
<td>string</td>
<td>required</td>
<td>field's id (obtained from the getAmountFields macro, only etc field is supported)</td>
<td>2020.1</td>
</tr>
</tbody>
</table>

Errors

See the following table for details about the getAmountFieldValue macro errors:

<table>
<thead>
<tr>
<th>Error</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Error Code</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Required argument is missing or undefined</td>
</tr>
<tr>
<td>SSS_COLUMN_DOES_NOT_EXIST</td>
<td>Invalid fieldId</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>Invalid lineIndex argument</td>
</tr>
</tbody>
</table>

## Work Breakdown Structure `getAmountFieldValue` Macro Syntax

The following code sample returns an object:

```javascript
{
    field: '01_2020_cost_etc',
    value: 25.5
}
```

The following code sample returns the ETC amount for the first line:

```javascript
require(['N/currentRecord'], function (currentRecord){
    var record = currentRecord.get();

    var etcCost = record.executeMacro({
        id: 'getAmountFieldValue'
        params: {
            lineIndex: 0,
            fieldId: '01_2020_cost_etc'
        }
    });
});
```

## `getUnmatchedActuals`

**Corresponding UI Button**  
Get Unmatched Actuals

**Macro Description**  
Gets unmatched actuals.

**Returns**  
Objects

**Supported Script Types**  
Client and server-side scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](https://www.oracle.com/).

**Since**  
2020.1

## Parameters

The `getUnmatchedActuals` macro does not require any additional parameter.

## Work Breakdown Structure `getUnmatched Actuals` Macro Syntax

The following code sample returns an object:

```javascript
{
    cost: 800
}
```
The following code sample returns unmatched actual cost and actual revenue:

```javascript
require(['N/currentRecord'], function (currentRecord) {
    var record = currentRecord.get();

    var unmatchedActuals = record.executeMacro({});
});
```

Revenue Recognition Event

**Note:** This topic applies to all versions of SuiteScript.

The revenue recognition event record is available only when the Advanced Revenue Management feature is enabled. In the UI, you access this record at Lists > Accounting > Revenue Recognition Events > New.

The internal ID for this record is `billingrevenueevent`.

Revenue recognition events trigger the creation of revenue recognition plans. Each revenue recognition event must have a revenue recognition event type. Revenue recognition event types are not scriptable. To create a revenue recognition event type, go to Lists > Accounting > Revenue Recognition Event Types > New. You must create at least one revenue recognition event type before you can create a revenue recognition event.

For help working with this record in the UI, see the help topic Creating a Custom Revenue Recognition Event.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Supported Script Types

The revenue recognition event record is supported in client and server SuiteScript.

The user events are not supported.

Supported Functions

The revenue recognition event record is partially scriptable. It can be created, read, updated, deleted, and searched using SuiteScript. It cannot be copied or transformed.
Usage Notes

The fields of this record that must be used in combination are fully described in the Revenue Recognition Event section of the help topic for the UI. For details, see the help topic Creating a Custom Revenue Recognition Event.

Code Samples

The following sample shows how to create revenue recognition events and perform other basic tasks.

```javascript
// Create Record
var revRecEvent = record.create({
    type: record.Type.BILLING_REVENUE_EVENT
});
revRecEvent.setValue({
    fieldId: 'transactionline',
    value: 12    // Transaction Line ID
});
revRecEvent.setValue({
    fieldId: 'eventtype',
    value: 2     // Event Type internal ID
});
revRecEvent.setValue({
    fieldId: 'eventpurpose',
    value: 'FORECAST'
});
revRecEvent.setValue({
    fieldId: 'eventdate',
    value: new Date('12/31/2016')
});
revRecEvent.setValue({
    fieldId: 'quantity',
    value: 1
});
var recID = revRecEvent.save();

//Load Record
var rec = record.load({
    type: record.Type.BILLING_REVENUE_EVENT,
    id: recID
});
```

Revenue Recognition Plan

**Note:** This topic applies to all versions of SuiteScript.

This record is used to indicate the posting periods in which revenue should be recognized and the amount to be recognized in each period. Revenue plans are derived from revenue recognition rules. Each revenue element has a forecast plan and one or more actual plans. The actual revenue plans control the posting of revenue.

This record is part of advanced revenue management. To use advanced revenue management, the Accounting Periods feature and Advanced Revenue Management feature must be enabled.
For help working with this record in the UI, see the help topic Setup for Advanced Revenue Management. The internal ID for this record is **revrecplan**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The revenue recognition plan record is supported in server SuiteScript only.

The user events are not supported.

**Supported Functions**

The revenue recognition plan record is partially scriptable. It can be read, updated, and searched using SuiteScript. It cannot be created, copied, deleted, or transformed.

**Planned Revenue Sublist**

The internal id for this sublist is **plannedrevenue**. The sublist type is inline editor.

The following table lists the scriptable sublist fields associated with the Planned Revenue sublist.

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Internal ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amount</td>
<td>Decimal Number</td>
<td>amount</td>
</tr>
<tr>
<td>Planned Period</td>
<td>List/Record</td>
<td>plannedperiod</td>
</tr>
</tbody>
</table>

For help working with this record in the UI, see the help topic Revenue Recognition Plans.

**Revenue Recognition Schedule**

**Note:** This topic applies to all versions of SuiteScript.

A revenue recognition schedule indicates the posting periods in which revenue should be recognized, and the amount to be recognized in each period, for an item sale. A revenue recognition schedule is generated for any sales transaction item that has an associated revenue recognition template. The point at which a revenue recognition schedule is generated for an item sale depends upon the type of sales transaction and enabled features and preferences set in your account. The schedule could be generated when a transaction is first saved, when it is approved, or when it is billed. Revenue recognition schedules provide a basis for the generation of journal entries that record the impact of item sales. This record is available when the Revenue Recognition feature is enabled.
For help working with this record in the UI, see the help topic *Working with Revenue Recognition Schedules*.

Revenue recognition schedules are system-generated.

The internal ID for this record is `revrecschedule`.

See the *SuiteScript Records Browser* for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic *Working with the SuiteScript Records Browser* in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

Client SuiteScript is not supported for the revenue recognition schedule record. It is scriptable in server SuiteScript only. Also, user event scripts are not supported.

**Supported Functions**

The revenue recognition schedule record is partially scriptable. It can be read, updated, and searched using SuiteScript. It cannot be created, copied, deleted, or transformed.

**Revenue Recognition Template**

**Note:** This topic applies to all versions of SuiteScript.

A revenue recognition template indicates how revenue from associated items should be posted. For each template: you can select from a choice of standard terms or define your own custom terms, set the time period over which recognition occurs, define an offset to delay the start of recognition, and set up an initial amount to be recognized. This record is available when the Revenue Recognition feature is enabled.

For help working with this record in the UI, see the help topics *Revenue Recognition Template* and *Creating Revenue Recognition Templates*.

The internal ID for this record is `revrectemplate`.

See the *SuiteScript Records Browser* for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic *Working with the SuiteScript Records Browser* in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module
Supported Script Types

Client SuiteScript is not supported for this record. It is scriptable in server SuiteScript only. Also, user event scripts are not supported.

Supported Functions

The revenue recognition template record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Role

**Note:** This topic applies to all versions of SuiteScript.

A role is a defined access configuration. To set up and manage user access to your NetSuite account, you need to set up roles that can be assigned to users. Roles include sets of permissions for viewing and/or editing data. Roles and their permissions determine the pages that users can see in the NetSuite interface and the tasks that they can complete. Each role is associated with a center, a UI designed for a particular business area.

For help working with this record in the UI, see the help topic NetSuite Roles Overview.

The internal ID for this record is `role`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

This record is partially scriptable in both client and server SuiteScript.

This record is scriptable in client with browser console only. You cannot deploy any client script on this record.

Supported Functions

This record is not fully scriptable. Only search is permitted. The Role record is supported in all search APIs, excluding the duplicate search APIs.

Usage Notes

To work with this record, a user must have Manage Roles permissions, at Setup > Users/Roles > Manage Roles > Search.
When passing a role into SuiteScript 2.0 search functions, the internal ID is case-sensitive and must be capitalized. The search type is `Role = search.Type.ROLE`. This is unique to the search APIs.

Sales Role

- **Note:** This topic applies to all versions of SuiteScript.

Sales roles can be defined for each member of a sales team. To use this record the Team Selling feature must be enabled. To enable team selling, go to Setup > Company > Enable Features > CRM. Click the Team Selling checkbox, and then click **Save**.

For help working with this record in the UI, see the help topic [NetSuite Roles Overview](#).

The internal ID for this record is `salesrole`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

- **Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

### Supported Script Types

The sales role record is supported in client and server SuiteScript.

### Supported Functions

The sales role record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Sales Tax Item

- **Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic [Sales Taxes](#).

The internal ID for this record is `salestaxitem`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

- **Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
Supported Script Types

Client SuiteScript is not supported for the sales tax item record. It is scriptable in server SuiteScript only.

Supported Functions

The sales tax item record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Subsidiary

Note: This topic applies to all versions of SuiteScript.

A NetSuite OneWorld account enables you to manage data for a hierarchical structure of separate legal entities, or subsidiaries. This structure is organized as a tree that rolls up to a root, or top-level parent subsidiary. The root subsidiary is the highest-level subsidiary in your account, and all other subsidiaries are below it in the hierarchy. If an account that is upgraded to OneWorld has preexisting data, this data is used for the root subsidiary.

Each subsidiary represents a separate company within your global organization. Subsidiaries can be international or domestic. When you create a subsidiary record in NetSuite, the country you define for its address determines the NetSuite edition and tax nexus associated with that subsidiary.

For help working with this record in the UI, see the help topics Subsidiaries in OneWorld and Subsidiary Setup.

The internal ID for this record is subsidiary.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The subsidiary record is supported in client and server SuiteScript.

Supported Functions

The subsidiary record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.
Usage Notes

The Company Information page and the root Subsidiary record share information. In OneWorld accounts, if you update shared fields on the Company Information page, the corresponding fields on the root Subsidiary record are also updated. An update to the Company Information page also triggers all user event scripts deployed on the Subsidiary record. If your script updates a shared field on the Subsidiary record, the corresponding field on the Company Information page is also updated.

The Subsidiary record is hidden in accounts that are not OneWorld, but the record still exists. You cannot deploy a stand-alone user event script on the Subsidiary record in accounts that are not OneWorld. You can, however, bundle your script and distribute it to these accounts. In this scenario, updates to the Company Information page trigger the execution of the bundled script.

You can create up to 124 subsidiary records in addition to the root subsidiary, for a total of 125.

On some records, the Subsidiary field is a multi-select field type. When executing a search containing multi-select fields, SuiteScript treats commas as a delimiter of values. It is therefore recommended that you avoid using commas in subsidiary record names. If you do not, subsidiary record names containing a comma can skew your search results.

Subsidiary Settings

Note: This topic applies to all versions of SuiteScript.

Some subsidiary-specific features require additional setup before you can use them. The Subsidiary Settings Manager page enables you to manage these subsidiary-specific features that require additional setup. After you enable a subsidiary-specific feature on the Enable Features page, that feature is available on the Subsidiary Settings Manager page.

From the Subsidiary Settings Manager page, you can access the Subsidiary Settings page for a specific subsidiary. On this page you can enable subsidiary-specific features and define their additional settings for this subsidiary. For information about using the Subsidiary Settings Manager page and its Subsidiary Settings pages, see the help topic Subsidiary Settings Manager.

The internal ID for this record is subsidiarysettings.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

Supported Script Types

The Subsidiary Settings record is scriptable in server SuiteScript. All server-side script types are supported.

Supported Operations

The Subsidiary Settings record can be read, updated, and searched using SuiteScript. However, searching is available only through SuiteAnalytics Workbook by searching for the Subsidiary Settings record type. For information about SuiteAnalytics Workbook, see the help topic SuiteAnalytics Workbook Overview.

The System Notes subtab on the Subsidiary Settings record is search-only. For information about the System Note record, see System Note.
Usage Notes

Your user role must have the Subsidiary Settings Manager edit permission to use the Subsidiary Settings record. For information about this permission, see the help topic Subsidiary Settings Manager Permissions.

⚠️ Important: The Subsidiary Settings page has a one-to-one relationship with its subsidiary record. When you create a subsidiary, NetSuite creates a Subsidiary Settings page for that subsidiary. When you delete a subsidiary, NetSuite deletes its corresponding Subsidiary Settings page.

The Subsidiary Settings record has the same Internal ID and External ID as its corresponding subsidiary record. For information about the Subsidiary record, see Subsidiary.

Code Sample

The following sample shows how to enable the Period End Journal Entries feature for a specific subsidiary.

```javascript
var parentSubsidiarySettings = record.load({
    type: 'subsidiarysettings',
    id: 1
});

// Enable the Period End Journal Entries feature
parentSubsidiarySettings.setValue({
    fieldId: 'allowperiodendjournalentries',
    value: true
});

// Set some settings for the feature
parentSubsidiarySettings.setValue({
    fieldId: 'createbscloseandopenjournals',
    value: true
});

parentSubsidiarySettings.setValue({
    fieldId: 'balancesheetopeningaccount',
    value: 45
});

parentSubsidiarySettings.setValue({
    fieldId: 'balancesheetclosingaccount',
    value: 46
});

var recId = parentSubsidiarySettings.save();
```

Supply Chain Snapshot

ℹ️ Note: This topic applies to all versions of SuiteScript.

The internal ID for this record is supplychainsnapshot.

The Supply Chain Snapshot record is available only when the following features are enabled:
Lists

- Multi-Location Inventory
- Advanced Inventory Management
- Demand Planning
- Supply Chain Control Tower

Use the Supply Chain Control Tower feature in your NetSuite OneWorld account to simulate inventory supply and demand across your supply chain. These simulations, or Snapshots, are used to analyze whether inventory levels are in line with demand or planned levels. This feature can help you to juggle how to match customer requests with supply availability.

For help working with this record in the UI, see the help topic [Supply Chain Control Tower](https://www.netsuite.com/help/topic/su/Supply-Chain-Control-Tower.html).

See the [SuiteScript Records Browser](https://www.netsuite.com/help/topic/su/SuiteScript-Records-Browser.html) for all internal IDs associated with this record. See the help topic [Usage Notes](https://www.netsuite.com/help/topic/su/Usage-Notes.html) for additional details about scriptable elements for this record.


See [Supply Chain Snapshot Record Actions](https://www.netsuite.com/help/topic/su/Supply-Chain-Snapshot-Record-Actions.html) for actions associated with this record. For more information about actions and macros, see the help topic [Overview of Record Action and Macro APIs](https://www.netsuite.com/help/topic/su/Overview-of-Record-Action-and-Macro-APIs.html).

**Supported Script Types**

The Supply Chain Snapshot record is scriptable in both client and server SuiteScript.

**Supported Operations**

The Supply Chain Snapshot record can be created, read, edited, deleted, and searched using SuiteScript. It cannot be copied.

**Usage Notes**

Supply Chain Snapshot details are based on a saved search. The record is scriptable in the same way that saved searches are scriptable. Because it is a snapshot of transactional data, you can refresh this view or run the snapshot details search as needed.

There are two types of saved searches that are associated with Supply Chain Snapshots:

- Supply chain snapshot details search – This is the line-by-line information as displayed by the snapshot record.
- Supply chain snapshot search – This is header-level information such as replenishment method, future horizon date, and past horizon date.

Field values on the record header can be set using scripts. Some fields are read-only or have special behaviors. Please see the table below for details.

The following are applicable elements of the Supply Chain Snapshot record:

**Fields**

<table>
<thead>
<tr>
<th>Field Id</th>
<th>Field Label</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memo</td>
<td>Memo</td>
<td>textarea</td>
<td>A typical memo field that can be set on create or edit.</td>
</tr>
<tr>
<td>Field Id</td>
<td>Field Label</td>
<td>Type</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>item</td>
<td>Item Name / Number</td>
<td>select</td>
<td>On create, value is selected or passed in as parameter. On edit, snapshot record is tied to item id, so the item field is read-only.</td>
</tr>
<tr>
<td>replenishmentmethod</td>
<td>Replenishment Method</td>
<td>select</td>
<td>This field is read-only. Its value is derived from the item record.</td>
</tr>
<tr>
<td>daterun</td>
<td>Date Run</td>
<td>datetime</td>
<td>This field is read-only. If refresh = true, it is the datetime stamp from when the data on the record was last refreshed. If refresh = false, it has not been updated.</td>
</tr>
<tr>
<td>pasthorizon</td>
<td>Past Horizon</td>
<td>select</td>
<td>Can be set on create. If refresh = T, can be set on edit. Options (in days): 7, 14, 30, 45, 60 Defaults to the value set in Inventory Management Preferences.</td>
</tr>
<tr>
<td>futurehorizon</td>
<td>Future Horizon</td>
<td>integer</td>
<td>This field is read-only. Its value is derived from the item record. If the item record value is empty, it derives its value from the value set in the Inventory Management Preferences. The value must be between 0 and 365 days.</td>
</tr>
<tr>
<td>futurehorizondate</td>
<td>Future Horizon Date</td>
<td>date</td>
<td>This field is read-only. It is calculated as follows: current date + the number of days set in the Past Horizon field.</td>
</tr>
<tr>
<td>refresh</td>
<td>Refresh Summary</td>
<td>checkbox</td>
<td>This is a non-persisted value. On create, this field cannot be set because data is fetched when the record is first created. On edit, this field determines whether to refresh the search results of snapshot details.</td>
</tr>
<tr>
<td>externalid</td>
<td>ExternalId</td>
<td>text</td>
<td>Hidden by default in the UI.</td>
</tr>
<tr>
<td>stockunit</td>
<td>Stock Unit</td>
<td>select</td>
<td>This field is read-only and shows only when the Multiple Units of Measure feature is enabled. Its value is derived from the value for Stock Units on the item record.</td>
</tr>
</tbody>
</table>

**Additional Notes**

- You can use the SuiteScript saved search API (the N/search module in SuiteScript 2.0 and nlapiCreateSearch(type, filters, columns) and nlapiLoadSearch(type, id) in SuiteScript 1.0) to script supply chain snapshots.
- Type is 'supplychainsnapshot' for searching general snapshot record info (header-level fields).
Type is 'supplychainsnapshotdetails' for searching line-level snapshot data.

Search is customizable. Searches of 'supplychainsnapshot' correspond to header fields, as shown in the table above. Searches of 'supplychainsnapshotdetails' access the transactional data in the details table, such as the following:

- item, subsidiary, location, date, sourcetype (Source), docnum (transaction), status, invbalance (Inventory Balance), demandqty, supplyqty, purchaseorderqty, salesorderqty, transferorderqty, workorderqty, otherdemandqty, othersupplyqty

- This above list includes labels in parentheses for potentially confusing column identifiers.

Scripts can be scheduled. Additional logic in these scripts is flexible. For example, it is possible to automatically analyze the inventory balance for an item to see if it is expected to be underwater, and then complete specific actions like creating purchase orders.

You can use an automated script for refreshing your supply chain snapshots. For example, every Sunday you could run a script to refresh all snapshots.

It is your responsibility to build, maintain, and export snapshot data to match your needs.

## Code Samples

The following sample shows how to create an item and set the future horizon to 10 days.

```javascript
var item = record.create({
  type: record.Type.INVENTORY_ITEM
});
item.setValue({
  fieldId: 'itemid',
  value: 'SS item test'
});
item.setValue({
  fieldId: 'taxschedule',
  value: '1'
});
item.setValue({
  fieldId: 'futurehorizon',
  value: '10'
});
var itemId = item.save();
```

The following sample shows how to generate snapshot for that item with past horizon of 14 days.

```javascript
var snap = record.create({
  type: record.Type.SUPPLY_CHAIN_SNAPSHOT
});
snap.setValue({
  fieldId: 'item',
  value: '7'
});
snap.setValue({
  fieldId: 'pasthorizon',
  value: '14'
});
var snapId = snap.save();
```

The following sample shows how to refresh the snapshot for an item with a past horizon of 7 days.
var loadSnap = record.load({
    type: record.Type.SUPPLY_CHAIN_SNAPSHOT,
    id: snapId,
    defaultValues: true
});
loadSnap.setValue({
    fieldId: 'refresh',
    value: 'T'
});
loadSnap.setValue({
    fieldId: 'pasthorizon',
    value: '7'
});
loadSnap.setValue({
    fieldId: 'memo',
    value: 'updated ss memo'
});
var recID = loadSnap.save();

The following sample shows how to create a saved search for a snapshot record.

var mySearch = search.create({
    type: search.Type.SUPPLY_CHAIN_SNAPSHOT,
    title: 'Snapshot header search for test item2',
    id: 'customsearch_scct',
    columns: [{
        name: 'item'
    }, {
        name: 'memo'
    }, {
        name: 'daterun'
    }],
    filters: [{
        name: 'item',
        operator: search.Operator.IS,
        values: itemId
    }]
});
var customsearch_scctId = mySearch.save();

The following sample shows how to create a saved search for snapshot details.

var mySearch = search.create({
    type: 'supplychainsnapshotdetails',
    title: 'Snapshot details search for test item2',
    id: 'customsearch_scct_details',
    columns: [{
        name: 'docnum'
    }, {
        name: 'location'
    }, {
        name: 'invbalance'
    }],
    filters: [{
        name: 'item',
    }]
});
The following sample shows how to load a search for snapshot record.

```javascript
var loadSearch = search.load({
  type: search.Type.SUPPLY_CHAIN_SNAPSHOT,
  id: 'customsearch_scct'
});
```

The following sample shows how to load search for snapshot details.

```javascript
var loadSearchDetails = search.load({
  type: 'supplychainsnapshotdetails',
  id: 'customsearch_scct_details'
});
```

Supply Chain Snapshot Record Actions

The supply chain snapshot record currently supports the refresh action.

For help working with this record in the UI, see the help topic Supply Chain Control Tower.

For information about SuiteScript 2.0 record actions, see the following help topics:

- Overview of Record Action and Macro APIs
- N/action Module

refresh

<table>
<thead>
<tr>
<th>Corresponding UI Button</th>
<th>Action Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>Takes a snapshot of the transactions executed for an item during a specified time period that includes past and future.</td>
<td>(notifications: [], response: {})</td>
<td>Client and server-side scripts</td>
<td>5 usage units</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Parameters

See the help topic `action.execute(options)` for details about parameters required for the execution of any action. The refresh action does not support any additional parameters.

Supply Chain Snapshot Refresh Action Syntax

```javascript
require(['N/action'], function(action){
```

SuiteScript Records Guide
System Note

Note: This topic applies to all versions of SuiteScript.

System notes track changes made to a record and to general configuration settings that have a financial impact. A system note for a change on a record captures the date when the change was made, who made the change, the role of the user who made the change, the type of change, and the old and new value in the record. System notes are logged on custom records as well as standard records. Anyone with view access to a record can view the system notes for that record. When searching system notes, you can search for certain records, use advanced search filters, create a saved search, or export the information for additional analysis.

For help working with this record in the UI, see the help topics System Notes Overview and Searching System Notes.

System notes are only exposed as a search.

In the UI, users who are not Administrators can only see system notes for which they are the authors. The same restriction applies to scripting.

The internal ID for this record is `systemnote`.

In SuiteScript 2.0, when using the `search.Type` enum, use the value `SYSTEM_NOTE`

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The system note record is scriptable in both client and server-side scripts.

Supported Functions

The system note record is partially scriptable. Only search scripting is permitted.

Code Samples

The following sample shows how you can script with system note records.

```javascript
require(['N/search'], function(search){
```

// action & action list loading
```
var actionList = action.find({recordType: 'supplychainsnapshot'});
var actionObj = action.get({recordType: 'supplychainsnapshot', id: 'refresh'})
```

// action execution
```
var result = actionObj.execute({recordId: 1});
var result = actionObj({recordId: 2});
var result = action.execute({recordType: 'supplychainsnapshot', id: 'refresh', params: {recordId: 1}});
```
function createSearch(){
    var systemNotesSearch = search.create({
        type: search.Type.SYSTEM_NOTE,
        title: 'Changes made in Employee Center',
        id: 'customsearch_emp_center_changes',
        columns: ['date', 'recordtype', 'record', 'field', 'oldvalue', 'newvalue'],
        filters: [{
            name: 'role',
            operator: search.Operator.IS,
            values: '15'
        }]
    });
    systemNotesSearch.save();
}
createSearch();

---

**Tax Control Account**

ℹ️ **Note:** This topic applies to all versions of SuiteScript.

A tax control account is an account to which the amounts computed for indirect taxes, such as sales tax and VAT, are posted.

In the UI, you can create a tax control account, and view existing ones, at Setup > Accounting > Tax Control Accounts. Note also that a tax control account is essentially an account record, with a few differences. Because they are accounts, tax control accounts also show up in the full list of accounts at Lists > Accounting > Accounts.

For help working with this record in the UI, see the help topic [Tax Control Accounts Overview](#).

The internal ID for this record is `taxacct`. See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

ℹ️ **Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

**Supported Script Types**

The tax control account record is scriptable in both client and server SuiteScript. However, user event scripts are not supported.

**Supported Functions**

The tax control account record is partially scriptable. Only creating, copying, and some updates are permitted.
Updating

Updating is permitted, but only of certain fields: Description, External ID, IsInactive, and Name. However, you may be able to change other fields when interacting with the tax control account as an account. For details, see Account.

Alternatives to Deleting

record.delete(options) is not supported, because a tax control account cannot be deleted (either through the UI or through SuiteScript). If the Advanced Taxes feature is enabled, you can effectively remove the tax control account by deleting the nexus that the account is associated with. If the Advanced Taxes feature is not enabled, you cannot remove the account.

An alternative to deleting the account is to make it account inactive, by setting the IsInactive field to true.

For more details on accounts that cannot be deleted in NetSuite, refer to the help topic Deleting Accounts and Making Accounts Inactive.

Field Definitions

This section describes some of the key fields on the tax control account.

Country

The value for country is derived from the nexus value and is read-only. If your account does not have the Advanced Taxes feature enabled, the value for country will always be the same for all your tax control accounts.

Description

The description of the record can include only up to 50 characters. Otherwise, the operation fails with an error reading “The field description contained more than the maximum number (50) of characters allowed.”

Description is one of the few fields that can be modified during an update.

External ID

ExternalID is one of the few fields that can be modified during an update.

IsInactive

Because a tax control account cannot be deleted, you might want to make it inactive.

IsInactive is one of the few fields that can be modified during an update.

Name

The name of the record must be unique. If you try to add a tax control account using a non-unique value for the name field, the system returns an error reading “This record already exists,” even if you included a unique external ID.

Name is one of the few fields that can be modified during an update.
Nexus

You can set a value for nexus only if the Advanced Taxes feature is enabled. If your NetSuite account does not use the Advanced Taxes feature, the value for nexus is set automatically and will always be the same for all your tax control accounts.

When Advanced Taxes is enabled, initialization of a nexus value is required. (For an example, refer to Code Samples.) However, after the account is created, the value for nexus cannot be changed.

Note that Advanced Taxes is enabled in all OneWorld accounts and cannot be turned off.

Tax Account Type

All tax control accounts have a tax account type, but you only actively choose a tax account type for accounts in certain countries. For example:

- If you are creating a tax code for the United Kingdom, valid choices include “sales” and “purchase.”
- When creating a tax code for the United States, only one option exists (“sales”), so you do not set a value for this field.

In countries where both “sales” and “purchase” are valid choices, the tax account type field is required. After an account has been created, the tax account type cannot be changed.

In countries where only one choice is allowed, the tax account type is automatically set, and you cannot change it through SuiteScript (or through the UI).

Note that if your account uses Advanced Taxes and has nexuses in many countries, you may need to set this field for some tax control accounts and not others.

For More Information

See the SuiteScript Records Browser for all internal IDs associated with this record.

Usage Notes

This section includes additional details on interacting with the tax control account record.

Finding the Internal ID

If you need the internal ID for an existing tax control account, note that you cannot find it through Setup > Accounting > Tax Control Accounts. However, you can find it at Lists > Accounting > Accounts. Make sure you have configured your NetSuite preference to “Show Internal IDs.” (You can find this choice at Home > Set Preferences.)

When sorting accounts at Lists > Accounting > Accounts, be aware that:

- Tax control accounts of type “sale” show up as Other Current Liability.
- Tax control accounts of type “purchase” show up as Other Current Asset.

Be aware that you can also add and update values for the external ID field.

Relationship to the Account Record

You can interact with the tax control account in many of the same ways you can interact with any account. However, some exceptions exist. For example, a tax control account cannot be deleted, and there are
limits to what you can update, as described in Updating. However, when you interact with the record as an account, you may be able to update additional fields.

For details on interacting with the account record, see Account.

**Code Samples**

The following samples show how you can script with tax control account records.

**Get**

The following sample shows how to retrieve a tax account record with the internal ID of ‘186’:

```javascript
var taxAcct = record.load({
  type: record.Type.TAX_ACCT,
  id: '186'
});
```

**Add**

The following sample shows how to create a tax control account when the Advanced Taxes feature is enabled. In this scenario, a value for nexus is required:

```javascript
var taxAcct = record.create({
  type: record.Type.TAX_ACCT,
  defaultValues: {
    'nexus': 'internal ID of nexus'
  }
});
taxAcct.setValue({
  fieldId: 'name',
  value: 'Name of account'
});
taxAcct.setValue({
  fieldId: 'description',
  value: 'Description of account'
});
taxAcct.setValue({
  fieldId: 'externalid',
  value: 'Your external ID value'
});
taxAcct.setValue({
  fieldId: 'taxaccttype',
  value: 'Type'
});
var recId = taxAcct.save();
```

**Update**

The following sample shows how to update the name, description, and external ID of a tax control account, and to make it active. Note that these fields are the only ones that can be updated.

```javascript
var taxAcct = record.load({
  type: record.Type.TAX_ACCT,
  id: '186'
});
taxAcct.setValue({
  fieldId: 'name',
  value: 'Updated name'
});
taxAcct.setValue({
  fieldId: 'description',
  value: 'Updated description'
});
taxAcct.setValue({
  fieldId: 'externalid',
  value: 'Updated external ID'
});
taxAcct.setValue({
  fieldId: 'active',
  value: true
});
var recId = taxAcct.save();
```
```
  type: record.Type.TAX_ACCT,
  id: 'internal ID of tax control account'
});
taxAcct.setValue({
  fieldId: 'name',
  value: 'Updated name'
});
taxAcct.setValue({
  fieldId: 'description',
  value: 'Updated description'
});
taxAcct.setValue({
  fieldId: 'isinactive',
  value: value
});
taxAcct.setValue({
  fieldId: 'externalid',
  value: 'Updated external ID of tax control account'
});
var recId = taxAcct.save();
```

Tax Group

**Note:** This topic applies to all versions of SuiteScript.

You can use the tax group record to combine several tax codes, even if the taxes are paid to different jurisdictions. For example, a tax group in the US might include a state tax, a city tax, and a transit tax. The advantage of using a tax group is that, when you create a sales invoice, you can apply one tax group to the transaction, instead of several separate tax codes.

In the UI, you navigate to this record by choosing Setup > Accounting > Tax Groups. For help working with this record in the UI, see the help topic Tax Groups Overview.

The internal ID for this record is `taxgroup`.

See the [SuiteScript Records Browser](https://www.oracle.com/netsuite/help/home/netsuite_customers.htm) for the internal IDs of fields, search filters, and search columns associated with this record.

For information about scripting with this record in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](https://www.oracle.com/netsuite/help/home/netsuite_customers.htm)
- [SuiteScript 2.0 Custom Forms](https://www.oracle.com/netsuite/help/home/netsuite_customers.htm)
- [N/record Module](https://www.oracle.com/netsuite/help/home/netsuite_customers.htm)

### Supported Script Types

The tax group record is scriptable in both client SuiteScript and Server SuiteScript. All three user events are supported: `beforeLoad`, `beforeSubmit`, and `afterSubmit`.

### Supported Functions

The tax group record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.
For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser.

## Code Samples

The following sample shows how to create a US tax group.

```javascript
var initValues = new Array();
initValues.nexuscountry = 'US';

var taxgroup = record.create({
  type: record.Type.TAX_GROUP,
  isDynamic: true,
  defaultValues: {
    'nexuscountry': 'US'
  }
});
taxgroup.setValue({
  fieldId: 'itemid',
  value: 'Test US Tax Group'
});
taxgroup.setValue({
  fieldId: 'description',
  value: 'Tax group description'
});
taxgroup.selectNewLine({
  sublistId: 'taxitem'
});
taxgroup.setCurrentSublistValue({
  sublistId: 'taxitem',
  fieldId: 'taxname',
  value: '-1425'        // -1425 is ID of TX_BUFFALO Tax Code
});
taxgroup.commitline({
  sublistId: 'taxitem'
});
recId = taxgroup.save();
```

The following code sample shows how to create a tax group for Canada.

```javascript
var newgroup = record.create({
  type: record.Type.TAX_GROUP,
  defaultValues: {
    'nexuscountry': 'CA'
  }
});
newgroup.setValue({
  fieldId: 'itemid',
  value: 'CA-T5'
});
newgroup.setValue({
  fieldId: 'piggyback',
  value: 'T'
});
newgroup.setValue({
```
Lists

```javascript
fieldId: 'taxitem1',
value: 21           // Canadian tax code
});
newgroup.setValue({
    fieldId: 'taxitem2',
    value: 24           // Canadian tax code
});
newgroup.setValue({
    fieldId: 'subsidiary',
    value: 2            // Canadian subsidiary
});
newgroup.setValue({
    fieldId: 'state',
    value: 'AB'         // Canadian subsidiary
});
newgroup.setValue({
    fieldId: 'description',
    value: 'New Canada Tax Group'
});
var recID = newgroup.save();
```

---

**Tax Period**

**Note:** This topic applies to all versions of SuiteScript.

A tax period defines a period of time over which your company tracks tax. A tax period can be a month, a quarter, or a year, depending on the frequency of your tax submissions or what your business has agreed with the local tax agency. You can run monthly, quarterly, or annual VAT/GST reports in NetSuite.

For help working with this record in the UI, see the help topic Tax Periods Overview.

The internal ID for this record is **taxperiod**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The tax period record is scriptable in server SuiteScript only.

**Supported Functions**

The tax period record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.
Tax Type

Note: This topic applies to all versions of SuiteScript.

A tax type determines where the tax paid or collected is tracked on the balance sheet. The balance sheet account to which NetSuite posts the collection or payment of tax is called the tax control account.

In NetSuite, the tax types are either already set up by default, or set up for you by Professional Services. When you add a subsidiary in a new country, the tax types for that country become available in the system automatically. However, an administrator can create new tax types if necessary.

For help working with this record in the UI, see the help topic Tax Types Overview.

The internal ID for this record is taxtype.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The tax type record is scriptable in server SuiteScript only.

Supported Functions

The tax type record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

Term

Note: This topic applies to all versions of SuiteScript.

Terms are used to specify when payment is due on your customers' invoices. Define the specific requirements of a term of payment by creating a term record. You can create different payment terms for different customers. In the UI, this is a user defined list at Setup > Accounting > Setup Tasks > Accounting Lists > Term > New.

For help working with this record in the UI, see the help topic Creating Terms of Payment.

The internal ID for this record is term. Search is not available on this record type.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
Lists

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The term record is scriptable in both client and server SuiteScript.

**Supported Functions**

The term record is partially scriptable. It can be created, updated, copied, and deleted using SuiteScript. It cannot be searched.

**Unit of Measure**

**Note:** This topic applies to all versions of SuiteScript.

The Multiple Units of Measure feature enables you to define various units used to stock, purchase, and sell inventory items, and track non-monetary accounts.

For help working with this record in the UI, see the help topics Multiple Units of Measure and Setting Up Units of Measure.

The internal ID for this record is `unitstype`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The unit of measure record is scriptable in server SuiteScript only.

**Supported Functions**

The unit of measure record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

**Vendor Category**

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic Vendor Records Overview.
The internal ID for this record is **vendorcategory**. Search is not available on this record type.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The vendor category record is scriptable in client and server SuiteScript.

**Supported Functions**

The vendor category record is fully scriptable, which means that it can be created, updated, copied, deleted, and searched using SuiteScript.

**Vendor-Subsidiary Relationship**

**Note:** This topic applies to all versions of SuiteScript.

If you use NetSuite OneWorld, you can share the vendor record with multiple subsidiaries, and then select those subsidiaries on core transactions.

For help working with Multi-Subsidiary Vendors in the UI, see the help topic Assigning Subsidiaries to a Vendor.

The internal ID for this record is **vendorsubsidiaryrelationship**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For customer-subsidiary and vendor-subsidiary relationships belonging to the same entity, both share the external ID of the entity. For example, Company ABC can be both a vendor and customer. If Company ABC has a vendor and customer relationship with the US subsidiary, both share the same external ID.

**Supported Script Types**

The vendor-subsidiary relationship is scriptable in both client and server SuiteScript.

All three user events are supported: **beforeLoad**, **beforeSubmit**, and **afterSubmit**.

**Supported Functions**

The vendor-subsidiary relationship can be read, created, updated, and deleted.
Code Sample

The following sample shows how to create a vendor-subsidiary relationship.

```javascript
var vsr = record.create({
    type: record.Type.VENDOR_SUBSIDIARY_RELATIONSHIP
});
vsr.setValue({
    fieldId: 'entity',
    value: 2
});
vsr.setValue({
    fieldId: 'subsidiary',
    value: 6
});
var recId = vsr.save();
```

Workplace

**Note:** This topic applies to all versions of SuiteScript.

To use SuitePeople U.S. Payroll, you must create workplace records for each location where you have a tax ID and maintain paid employees. NetSuite uses workplace records to define the taxes the employer must withhold and pay. Each employee you include in a payroll must have a designated workplace.

For help working with this record in the UI, see the help topic Entering Workplace Records for Payroll.

This record is available only when the Payroll feature is enabled at Setup > Company > Enable Features on the Employees subtab.

In the UI, you can access this record at Lists > Employees > Workplaces.

The internal ID for this record is `workplace`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

The workplace record is scriptable in both client and server SuiteScript.

**Supported Functions**

The workplace record is fully scriptable, which means that the record can be created, updated, copied, deleted, and searched using SuiteScript.
Customization

Note: This topic applies to all versions of SuiteScript.

The following customization records are scriptable in SuiteScript:

- Custom List
- Custom Record
- Custom Segment Fields and Values
- Custom Transaction
- Scheduled Script Instance
- Script (individual script types)
- Script Deployment

Custom List

Note: This topic applies to all versions of SuiteScript.

A custom list is a list of values that you can use in custom fields on your forms and records. Custom lists enable you to set up predefined choices for your employees and customers to select when entering transactions and records. Custom lists can be searched using search filters and search columns.

For help working with this record in the UI, see the help topic Custom Lists.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with custom lists, see the following help topics:

- SuiteScript 2.0 Custom List Pages
- Custom List IDs
- Search Filters
- Search Columns

Custom List IDs

Each custom list has a unique ID. This value shown in the ID field on the custom list record. When creating or interacting with an instance of a custom record type, you must use this ID.

Every custom list ID is prefaced by customlist. If the ID was entirely system generated, it ends with a number (for example, customlist100). If the ID was customized when the record was created, the ID may be more descriptive. To view a list of all these IDs, navigate to Customization > Lists, Records, & Fields > Lists. These values are shown in the ID column.

Search Filters

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Field Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>internalid</td>
<td>Internal ID</td>
<td>select</td>
</tr>
<tr>
<td>internalidnumber</td>
<td>Internal ID (Number)</td>
<td>integer</td>
</tr>
</tbody>
</table>
A custom record can be used to collect information specific to the needs of your business. For example, you may want to keep track of training courses your employees have taken. Since a record type specific for this purpose does not exist in NetSuite, you could create a custom record type, titled Employee Courses, specifically to store the training course data.

The NetSuite UI enables you to create custom record types and work with instances of those types. You cannot use SuiteScript to create a custom record type or make changes to an existing custom record type. However, you can use SuiteScript to interact with instances of an existing custom record type.

For example, suppose your system had a custom record type called Feature. Each instance of that record type would be a feature record. You could use SuiteScript to update an existing feature record, delete a feature record, or create a new feature record. However, you could not use SuiteScript to alter the Feature custom record type.

For help working with this record in the UI, see the help topic Creating Custom Record Types.

See the SuiteScript Records Browser for all internal IDs associated with this record, which includes a list of fields commonly available on instances of all custom records.

To interact with fields that were created for a specific record custom type, use the IDs for those fields. These IDs have a prefix of custrecord. For help finding these IDs, see the help topic How do I find a field's internal ID?

For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with records in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- N/record Module

Custom Record IDs

Each custom record has a unique ID. This value is shown in the ID field on the custom record. When creating or interacting with an instance of a custom record type, you must use this ID.
Every custom record ID is prefaced by `customrecord`. If the ID was entirely system generated, it ends with a number (for example, `customrecord100`). If the ID was customized when the record was created, the ID may be more descriptive. To view a list of all IDs, navigate to Customization > Lists, Records, & Fields > Record Types. The values are shown in the ID column.

You can also see the ID on the record that represents the custom record type.

In SuiteScript 2.0, you use this value to set the `Record.type` or `CurrentRecord.type` field. Note that this guidance differs from the way you set this field if you are working with a standard NetSuite record type. When working with a standard record type, you set this field by using either the `record.Type` or `CurrentRecord.type` enum.

See the Code Sample for examples.

**Supported Script Types**

Custom records (instances of a custom record type) are scriptable in both client and server SuiteScript.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

Custom records are fully scriptable — they can be created, updated, copied, deleted, and searched using SuiteScript.

**Code Sample**

In SuiteScript 2.0, you use the `N/record Module` and `N/currentRecord Module` to interact with custom records, just as you would with a standard record type. Note that you use the custom record type's string
ID when setting the `Record.type` field, as shown in the following example. For help finding this ID, see Custom Record IDs.

The following sample shows how to work with instances of a custom record type with the script ID `customrecord_feature`. This sample assumes that this custom record type has fields with the field IDs `custrecord_priority` and `custrecord_risklevel`.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */
define([ 'N/record' ], function(record) {
    function afterSubmit (context) {
        // Use a string internal ID to identify the custom record type.
        var rec = record.create({
            type: 'customrecord_feature',
            isDynamic : true
        });

        rec.setValue({
            fieldId: 'name',
            value: 'entry form redesign'
        });

        rec.setValue({
            fieldId: 'custrecord_priority',
            value: 1
        });

        rec.setValue({
            fieldId: 'custrecord_risklevel',
            value: 3
        });

        try {
            var callId = rec.save();
            log.debug('Call record created successfully', 'Id: ' + callId);
        } catch (e) {
            log.debug('sigh');
            log.error(e.name);
        }
    }
    return {
        afterSubmit: afterSubmit
    };
});
```

Custom Segment Fields and Values

**Note:** This topic applies to all versions of SuiteScript.

Custom segments are custom classification fields similar to class, department, and location. When the Custom Segments feature is enabled, an administrator working in the UI can create custom segments,
define possible values for each segment, and apply the segments to specific record types. A custom segment appears as a list or multi-select field on instances of record types where the segment was applied. Users can classify records by selecting the appropriate value for each segment.

For help working with this record in the UI, see the help topic Custom Segments.

You cannot use SuiteScript to create a custom segment. However, you can use SuiteScript to create values for existing custom segments. You can also use SuiteScript to set values for segments that appear as fields on record instances. For details, see the following:

- Using SuiteScript to Create Values for Existing Custom Segments
- Using SuiteScript to Set Values for Custom Segment Fields

Using SuiteScript to Create Values for Existing Custom Segments

Your NetSuite account contains a custom record type for every custom segment that has been defined. To add a value to the custom segment, you add an instance of the corresponding custom record type.

You can find the internal ID of the appropriate custom record type on the custom segment definition, under the Record ID label.

For help working with this record in the UI, see the help topic Custom Segments.

Code Sample

The following sample shows how to create a value for a segment called Profit Center.

```javascript
// Create new segment value
var newSegmentValue = record.create({
    type: 'customrecord_cseg_profitcenter'
});

// Set a name value on segment value record
newSegmentValue.setValue ({
    fieldId: 'name',
    value: stItemValue
});
```
// If appropriate, you may want to set a parent value for the new value.
// For example, you might have a static value that you use for all
// segment values created through a script.
newSegmentValue.setText ({
  fieldId: 'parent',
  text: 'Created from automated script'
});

...

Using SuiteScript to Set Values for Custom Segment Fields

When a segment is applied to a record type, it appears as a field on instances of that type. You can interact with this field as you would a custom field. To do this, you need to be able to identify both the segment and the relevant values.

As of 2018.2, a unified script ID is available for custom segment definitions. You can use a single ID to refer to a custom field related to a particular custom segment on any record type the segment is applied to. In this case, the script ID refers to a custom field and not to a particular record type. With the unified script ID, you do not need to distinguish among script ID values based on the record type you are working with.

For help working with this record in the UI, see the help topic Creating a Custom Segment.

For details, see the following topics:

- Custom Segments
- Identifying a Segment Field when Using the Legacy Script ID
- Identifying a Segment Field when Using the Unified Script ID
- Identifying a Segment Value
- Code Sample

Identifying a Segment Field when Using the Legacy Script ID

You identify a segment field in the same way you would a custom field: by using its field ID. To find the field ID, view an instance of the record type that you want to interact with. Click the label of the field to display a window that includes the field ID.

If you have permission to view custom segment definitions, you might notice that the segment definition lists a script ID and record ID. You should not refer to either of these IDs when setting a value for the segment on a record instance.
Additionally, be aware that a single custom segment can appear on instances of multiple record types. The field ID for the segment can vary among record types. You must use the field ID as it appears on an instance of the **same record type** where you want your script to set a segment value.

**Identifying a Segment Field when Using the Unified Script ID**

If you use the unified script ID for a custom segment, the script ID refers to all custom fields for record types that the segment is applied to. In this case, you must use the ID as it appears on the custom segment record to identify any custom field on a custom segment.

If the Use as Field ID box is checked, it means that you are using the unified script ID for the particular custom segment only.

**Identifying a Segment Value**

Typically, each custom segment includes a list of possible values. To set a value for a segment by using SuiteScript, at least one value must already have been defined for the segment.

You can reference a custom segment value by specifying either of the following:

- The value's text label
- The value's internal ID

The text label is shown in the custom segment definition and on instances of the record type where the segment has been applied. On the record instance, the segment’s values are shown in either a multi-select box or a list.

To find a segment value’s internal ID, open the segment definition at Customization > Lists, Records, & Fields > Custom Segments. To see a list of all possible values, refer to the Values sublist. This list includes a column labeled ID, which identifies each value’s internal ID.
The following snippet shows how to set a value for a segment with the field ID of `custbody_cseg_profitcenter`. This uses the legacy script ID to refer to a transaction body field on a custom segment.

```javascript
objRecord.setText({
    fieldId: 'custbody_cseg_profitcenter',
    text: 'Outerwear',
    ignoreFieldChange: false
});
```

**Custom Transaction**

**Note:** This topic applies to all versions of SuiteScript.

You use the custom transaction record to interact with instances of existing custom transaction types. For example, suppose you had a custom transaction type named Non-Operational Income Entry. In this case, you could use SuiteScript to create and modify non-operational income entries.

For help working with this record in the UI, see the help topic Custom Transactions.

The internal ID for a custom transaction record varies depending on the ID value of the custom transaction type. For example, if your custom transaction type had an ID value of `_noie`, the SuiteScript internal ID would be `customtransaction_noie`. You can view the custom transaction type's ID value by opening it editing (go to Customization > Lists, Records, & Fields > Transaction Types) and selecting the appropriate type.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with records in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- N/record Module

**Supported Script Types**

The custom transaction record (instance of a custom transaction type) are scriptable in both client and server SuiteScript.
All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

The custom transaction record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

You use the standard transaction search to search custom transactions. For more information, see [Transaction Search](#).

Custom transactions can be voided if configured to support the void option. Direct voids are never permitted with custom transactions. Only voids through reversal journals permitted. See the help topic `transaction.void(options)`.

**Usage Notes**

In the UI, custom transactions can have a Status field. In SuiteScript, this field has an internal ID of `transtatus`. Values for `transtatus` are set by the system when the statuses are created. They cannot be modified. Possible values are A through Z. In the UI, you can view a transaction type’s available statuses (and their `transtatus` values) by opening the transaction type and navigating to the Statuses subtab. The `transtatus` value for each status is listed in the Code column.

**Code Sample**

The following sample shows how to create an instance of a custom transaction type.

```javascript
var transaction = record.create ({
    type: 'customtransaction_mytype'
});
transaction.setValue ({
    fieldId: 'subsidiary',
    value: '1'
});
// Add debit line
transaction.selectNewLineItem ({
    sublistId: 'line'
});
transaction.setCurrentSublistValue ({
    sublistId: 'line',
    fieldId: 'account',
    value: '1'
});
transaction.setCurrentSublistValue ({
    sublistId: 'line',
    fieldId: 'debit',
    value: 10
});
transaction.setCurrentSublistValue ({
    sublistId: 'line',
    fieldId: 'credit',
    value: 0
});
```

---

*Oracle NetSuite*
```javascript
  sublistId: 'line',
  fieldId: 'memo',
  value: 'My first custom transaction line'
});
transaction.commitLine ({
  sublistId: 'line'
});

// Add credit line
transaction.selectNewLineItem ({
  sublistId: 'line'
});
transaction.setCurrentSublistValue ({
  sublistId: 'line',
  fieldId: 'account',
  value: '2'
});
transaction.setCurrentSublistValue ({
  sublistId: 'line',
  fieldId: 'debit',
  value: 0
});
transaction.setCurrentSublistValue ({
  sublistId: 'line',
  fieldId: 'credit',
  value: 10
});
transaction.setCurrentSublistValue ({
  sublistId: 'line',
  fieldId: 'memo',
  value: 'My second custom transaction line'
});

transaction.commitLine ({
  sublistId: 'line'
});

transaction.save();
```

## Scheduled Script Instance

**Note:** This topic applies to all versions of SuiteScript.

You can use the scheduled script instance search to return information about individual scheduled script and map/reduce script jobs. A subset of this information is displayed on the Scheduled Scripts and Map/Reduce Scripts pages in the NetSuite UI:

- **Scheduled Scripts:** Each entry on the Scheduled Script Status page contains information about a single scheduled script instance. Each scheduled script instance is handled by one scheduled script job. To access the Scheduled Script Status page in the UI, go to [Scheduled Scripts](#).
- **Map/Reduce Scripts:** Each entry on the Map/Reduce Script Status page contains information about a single map/reduce script instance. Each map/reduce script instance is handled by multiple map/reduce script jobs. To access the Map/Reduce Script Status page in the UI, go to [Map/Reduce Scripts](#). You can access information about the individual map/reduce jobs for a particular map/reduce script instance from [Map/Reduce Records Guide](#).
the map/reduce script details page. To access this page from the Map/Reduce Script Status page, click Details for the appropriate script instance.

For more information on the Scheduled Script Status page, see the help topic SuiteScript 2.0 Scheduled Script Status Page. For more information on the Map/Reduce Script Status page, see the help topic SuiteScript 2.0 Map/Reduce Script Status Page.

For help working with this record in the UI, see the help topic Monitoring a Scheduled Script’s Runtime Status.

The internal ID for this record is scheduledscriptinstance.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with records in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- N/record Module

### Supported Script Types

The scheduled script instance record is scriptable in server SuiteScript only.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

### Supported Functions

The scheduled script instance record is partially scriptable — it can only be searched. It cannot be created, updated, copied, or deleted using SuiteScript.

### Usage Notes

You can script searches on scheduledscriptinstance with all server script types. Scheduled script instance searches are not exposed to client scripts.

For a list of SuiteScript supported scheduledscriptinstance fields, see the SuiteScript Records Browser. In addition, see the following table for supplemental information on three supported fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Priority</td>
<td>This field holds the original priority setting (low, standard, or high) for the scheduled script job or map/reduce job. The priority setting is the value set when the scheduled script job or map/reduce job is initially created.</td>
</tr>
<tr>
<td>(originalpriority)</td>
<td>- If a job is created when a deployment is submitted for processing, the original priority is the priority set on the deployment record at the time it was submitted for processing.</td>
</tr>
<tr>
<td></td>
<td>- If a job is created during script processing, the original priority is the current priority of the job that created the new job.</td>
</tr>
</tbody>
</table>

For more information on deployment record priority settings, see the help topic SuiteCloud Processors – Priority Levels.
Note: For scheduled scripts, one scheduled script job is created when a script deployment is submitted for processing.

For map/reduce scripts, the initial map/reduce jobs are created when the script deployment is submitted for processing. In addition, more map/reduce jobs can be created during script processing.

<table>
<thead>
<tr>
<th>Field</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Priority Number</td>
<td>This field holds a numeric value that corresponds to the original priority value:</td>
</tr>
<tr>
<td>(originalPriorityNumber)</td>
<td>■ High priority: 1</td>
</tr>
<tr>
<td></td>
<td>■ Standard priority: 2</td>
</tr>
<tr>
<td></td>
<td>■ Low priority: 3</td>
</tr>
<tr>
<td></td>
<td>You cannot group your search results by the original priority field because it holds a text value. The original priority number field is provided as an alternative.</td>
</tr>
<tr>
<td>Priority Timestamp</td>
<td>This field holds the time that the current priority is set for the scheduled script job or map/reduce job. This field can reflect the timestamp of a user-defined priority setting change or a system-defined priority setting change. User-defined priority changes do not impact the value of this field for existing jobs. However, system-defined priority changes may impact this value for existing jobs. For example, if you create a scheduled script deployment and immediately submit the deployment for processing, the priority timestamp for that job reflects the deployment submission time. If you then change the priority on the deployment record, neither the priority nor the priority timestamp for the existing job is changed. However, If you enable priority elevation, and the priority for the existing job is changed by the system, the priority timestamp value is changed accordingly. For more information on priority elevation, see the help topic SuiteCloud Processors – Priority Elevation and Processor Reservation (Advanced Settings).</td>
</tr>
<tr>
<td>(prioritytimestamp)</td>
<td></td>
</tr>
</tbody>
</table>

**Code Sample**

The following sample shows how to perform a search for all schedule script instance records that are in progress in queue 2. It uses search filters and columns to return all available information on the filtered results. Note that the first two search columns are joins on the script and script deployment record types.

```javascript
var searchResults = search.create ({
  type: 'scheduledscriptinstance',
  filters: [
    {name: 'queue',
      operator: IS,
      values: '2'},
    {name: 'status',
      operator: ANYOF,
      values: 'PROCESSING'},
  ],
  columns: [
    {name: 'name', join: 'script'},
    {name: 'internalid', join: 'scriptdeployment'},
    {name: 'datecreated'},
    {name: 'status'},
    {name: 'startdate'},
  ],
});```
Script

**Note:** This topic applies to all versions of SuiteScript.

A script record allows you to deploy your entry point script. The SuiteScript Records Browser lists the script record for each individual script type separately (see table below).

For help working with script records in the UI, see the help topic [Script Record Creation](#).

Each script record has a unique internal ID. The following table lists each internal id and provides links to the SuiteScript Records Browser for each individual type of script record. Refer to these records for all other internal IDs associated with each record.

<table>
<thead>
<tr>
<th>Script Record</th>
<th>Internal ID</th>
<th>SuiteScript Records Browser Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bundle Installation Script</td>
<td>bundleinstallationscript</td>
<td>Bundle Installation Script</td>
</tr>
<tr>
<td>Client Script</td>
<td>clientscript</td>
<td>Client Script</td>
</tr>
<tr>
<td>Map/Reduce Script</td>
<td>mapreducescript</td>
<td>Map/Reduce Script</td>
</tr>
<tr>
<td>Massupdate Script</td>
<td>massupdatescript</td>
<td>Massupdate Script</td>
</tr>
<tr>
<td>Portlet</td>
<td>portlet</td>
<td>Portlet</td>
</tr>
<tr>
<td>Restlet</td>
<td>restlet</td>
<td>Restlet</td>
</tr>
<tr>
<td>Suitelet</td>
<td>suitelet</td>
<td>Suitelet</td>
</tr>
<tr>
<td>User Event Script</td>
<td>usereventscript</td>
<td>User Event Script</td>
</tr>
<tr>
<td>Workflow Action Script</td>
<td>workflowactionscript</td>
<td>Workflow Action Script</td>
</tr>
</tbody>
</table>

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with records in SuiteScript, see the following help topics:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [N/record Module](#)

### Supported Script Types

Script records are scriptable in server SuiteScript only.
All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

**Supported Functions**

Each script record is partially scriptable — it can be updated and searched. It cannot be created, copied, or deleted.

**Usage Notes**

You can load, update and save, and search script records with all server script types. Script records are not exposed to client scripts.

The following objects are not scriptable on any script record:
- The Change ID button (in Edit view of UI)
- The Unhandled Errors tab
- The History tab
- The System Notes tab
- The Execution Log tab
- The Libraries tab

The following objects are read-only (editing is not supported):
- The Type field
- The Parameters tab

To access script records within a script, the script owner must belong to a role assigned with SuiteScript permissions.

> **Important:** Scripts can only access script records when the records are part of the same bundle. Scripts that are not part of a bundle cannot access script records that are part of a bundle.

For the id parameter in `record.load(options)`, `search.load(options)`, or `search.create(options)` methods, you must use the script record's internal ID (for example, 191); the script record's ID (for example, `customscript_csalert`) is not supported. You can find a script record's internal ID at Customization > Scripting > Scripts. If you have the Show Internal IDs preference enabled at Home > Set Preferences, internal IDs are listed in the Internal ID column.

For record and search methods that require the type parameter, use one of the following `record.Type` or `search.Type` values:

- **BUNDLE_INSTALLATION_SCRIPT**
Code Samples

The following samples show how to load and read a script record, how to create a script record, and how to search a script record.

```javascript
// Read a SUITELET script record
var rec = record.load({
  type: record.Type.SUITELET,
  id: 605
});
var a = rec.getValue({
  fieldId: 'scriptfile'
});
var count = rec.getLineCount({
  sublistId: 'parameters'
});
var pLabel = rec.getSublistValue({
  sublistId: 'parameters',
  fieldId: 'label',
  line: 1
});
var pId = rec.getSublistValue({
  sublistId: 'parameters',
  fieldId: 'internalid',
  line: 1
});
var pType = rec.getSublistValue({
  sublistId: 'parameters',
  fieldId: 'fieldtype',
  line: 1
});
var pRecordType = rec.getSublistValue({
  sublistId: 'parameters',
  fieldId: 'selectrecordtype',
  line: 1
});

// Edit a user event script record
var rec = record.load({
  type: record.Type.USEREVENT_SCRIPT,
  id: 302
});
```
rec.setValue({
    fieldId: 'name',
    value: 'userevent_001'
});
rec.setValue({
    fieldId: 'scriptfile',
    value: '227'
});
rec.setValue({
    fieldId: 'notifyadmins',
    value: 'T'
});
rec.setValue({
    fieldId: 'aftersubmitfunction',
    value: 'afterSubmitFunction'
});
rec.save();

// Search a script record
var searchFilters = search.createFilter({
    name: 'defaultfunction',
    operator: search.Operator.IS,
    values: 'myfunctionname'
});
var searchColumns = search.createColumn({
    name: 'name'
});
var mySearch = search.create({
    type: search.Type.SUITELET,
    filters: searchFilters,
    columns: searchColumns
});
var mySearchResults = mySearch.run();

## Script Deployment

**Note:** This topic applies to all versions of SuiteScript.

Script deployment records are required in order to run your script in your NetSuite account.

For help working with this record in the UI, see the help topic Script Deployment.

The internal ID for the script deployment record is **scriptdeployment**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with records in SuiteScript, see the following help topics:

- SuiteScript 2.0 Scripting Records and Subrecords
- N/record Module
Supported Script Types

Script records are scriptable in server SuiteScript only.

Supported Functions

The script deployment record is fully scriptable — it can be created, copied, updated, deleted, and searched using SuiteScript.

Usage Notes

You can access script deployment records with all server script types. Script deployment records are not exposed to client scripts.

The following objects are not scriptable on the script deployment record:

- Change ID button (in Edit view of UI)
- The History subtab
- The System Notes subtab
- The Execution Log subtab
- The Scheduling subtab

The following object is read-only on the script deployment record (editing is not supported):

- The Parameters subtab

To access script deployment records within a script, the script owner must belong to a role assigned with SuiteScript permissions.

**Important:** Scripts can only access script deployment records when the records are part of the same bundle.

Scripts that are not part of a bundle cannot access script deployment records that are part of a bundle.

You can use script deployment records to set up context filtering. Context filtering lets you specify how and when the associated script runs. For more information, see the help topics Execution Contexts and Localization Context.

For methods that require the ID parameter, you must use the script deployment record's internal ID (for example, 88). The script deployment record's script ID (for example, customdeploy_newsfeed_email) is not supported. You can find a script deployment record's internal ID at Customization > Scripting > Script Deployments. If you have the Show Internal IDs preference enabled at Home > Set Preferences, internal IDs are listed in the Internal ID column.
For record and search methods that require the type parameter, use the `record.Type.SCRIPT_DEPLOYMENT` or `search.Type.SCRIPT_DEPLOYMENT` value.

**Code Samples**

The following samples show how to create a script deployment record, how to edit a script deployment record, how to read script deployment record, how to copy and edit a script deployment record, and how to search a script deployment record.

```javascript
// Create a script deployment record
var rec = record.create({
    type: record.Type.SCRIPT_DEPLOYMENT,
    defaultValues: {
        script: 2 //scriptId
    }
});

// Edit a script deployment record
var rec = record.load({
    type: record.Type.SCRIPT_DEPLOYMENT,
    id: 65
});
rec.setValue({
    fieldId: 'allroles',
    value: 'T'
});
rec.save();

// Read a script deployment record
var rec = record.load({
    type: record.Type.SCRIPT_DEPLOYMENT,
    id: 203
});
var recStatus = rec.getValue({
    fieldId: 'status'
});

// Copy and edit a script deployment record
var rec = record.copy({
    type: record.Type.SCRIPT_DEPLOYMENT,
    id: 41
});
rec.setValue({
    fieldId: 'isdeployed',
    value: 'F'
});
rec.setValue({
    fieldId: 'title',
    value: 'Feature 4141 - Suitelet Scratch 4'
});
rec.save();

// Search a script deployment record
```
```javascript
var searchFilters = search.createFilter({
  name: 'internalidnumber',
  operator: search.Operator.ANYOF,
  values: '43'
});
var searchColumns = search.createColumn({
  name: 'title'
});
var mySearch = search.create({
  type: search.Type.SCRIPT_DEPLOYMENT,
  filters: searchFilters,
  columns: searchColumns
});
var mySearchResults = mySearch.run();
```

Marketing

**Note:** This topic applies to all versions of SuiteScript.

The following marketing records are scriptable in SuiteScript:

- Campaign
- Campaign Response
- Campaign Template
- Coupon Code
- Email Template
- Promotion

Campaign

**Note:** This topic applies to all versions of SuiteScript.

Campaign records are used to manage your marketing initiatives. With campaign records, you can manage your online marketing as well as your offline efforts. You can create and send email marketing messages, manage paid keywords, send promotions through the mail, and manage your advertising campaigns.

For help working with this record in the UI, see the help topics Campaign Overview and Creating a Campaign Record.

The internal ID for this record is **campaign**.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:
Supported Script Types

The campaign record is scriptable in both client and server SuiteScript.

Supported Functions

The campaign record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Campaign Response

**Note:** This topic applies to all versions of SuiteScript.

The campaign response record helps you refine how you deliver marketing campaigns.

To view this record in the UI, navigate to an entity record and click the Marketing subtab. On the Campaigns subtab, click the Add Response button.

For help working with this record in the UI, see the help topic Tracking Campaign Responses.

The internal ID for this record is `campaignresponse`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- Tracking Campaign Responses
- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The campaign response record is scriptable in both client and server SuiteScript.

Supported Functions

The campaign response record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.
Campaign Template

Note: This topic applies to all versions of SuiteScript.

The campaign template record is used for both CRMSDK templates and scriptable templates. In the UI, you find this record by choosing Documents > Templates > Marketing Templates.

For help working with this record in the UI, see the help topic Email Marketing Campaigns.

Scriptable templates are not supported with SuiteScript.

The internal ID for this record is campaigntemplate.

See the SuiteScript Records Browser for the internal IDs of all fields associated with this record, their corresponding labels in the UI, and more details. Fields not listed in the Records Browser are not supported and should not be used.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The campaign template record is scriptable in server SuiteScript only.

All three user events are supported: beforeLoad, beforeSubmit, and afterSubmit.

Field Definitions

This section describes some of the key fields on the campaign template record.

Content and Mediaitem

When accessing this record, only one of the following fields contains a value:

- content — This field maps to the text editor that appears on the Template tab in the UI. This field contains plain text or HTML.
- mediaitem — This field maps to the File text box that appears on the Template tab in the UI. This field identifies the file that is used as the basis for the template.

Code Sample

The following sample shows how to access template content:

```javascript
var template = record.load({
    type: record.Type.CAMPAIGN_TEMPLATE,
});
```
Coupon Code

**Note:** This topic applies to all versions of SuiteScript.

Promotions allow you to track the source of revenue and to offer discounts in the form of coupons. Each promotion has a promotion code that can be applied to transactions and campaigns. Enable Promotion Codes on the Transactions subtab at Setup > Company > Setup Tasks > Enable Features (Administrator).

For help working with this record in the UI, see the help topic Selling With Promotion Codes.

The internal ID for this record is `couponcode`.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- **SuiteScript 2.0 Scripting Records and Subrecords**
- **SuiteScript 2.0 Custom Forms**
- **N/record Module**

**Supported Script Types**

The coupon code record is scriptable in both client and server SuiteScript.

**Supported Functions**

The coupon code record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

Email Template

**Note:** This topic applies to all versions of SuiteScript.

The email template record is used for both CRMSDK templates and scriptable templates. In the UI, you find this record by choosing Documents > Templates > Email Templates. This record is available only if the
Customer Relationship Management feature has been enabled at Setup > Company > Enable Features, on the CRM tab.

For help working with this record in the UI, see the help topic Working with Email Templates.

Scriptable templates are not supported with SuiteScript.

The internal ID for this record is `emailtemplate`.

See the SuiteScript Records Browser for the internal IDs of all fields associated with this record, their corresponding labels in the UI, and more details. Fields not listed in the Records Browser are not supported and should not be used.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

**Supported Script Types**

This record is only supported in server SuiteScript.

All three user events are supported: `beforeLoad`, `beforeSubmit`, and `afterSubmit`.

**Field Definitions**

This section describes some of the key fields on the email template record.

**Content and Mediaitem**

When accessing this record, only one of the following fields contains a value:

- content — This field maps to the text editor that appears on the Template tab in the UI. This field contains plain text or HTML.
- mediaitem — This field maps to the File text box that appears on the Template tab in the UI. This field identifies the file that is used as the basis for the template.

**Code Sample**

The following sample shows how to access template content:

```javascript
var template = record.load({
    type: record.Type.EMAIL_TEMPLATE,
    id: 124
});

if (template.getValue({fieldId: 'mediaitem'}) != null) {
    var media = template.getValue({fieldId: 'mediaitem'});
    // do something with the media...
```
Promotions allow you to create special offers to motivate your customers to purchase products in higher quantities. Promotions can help you to move out-of-date stock, increase sales, and reward valuable clients. Promotions provide discounts that customers can apply to Web store orders and orders placed with sales reps. Furthermore, NetSuite promotions enable you to target specific customers, locations or channels, and time periods.

For help working with this record in the UI, see the help topics Promotions Overview, Configuring Promotions, and The Promotion Record.

The internal ID for this record is promotioncode.

See the SuiteScript Records Browser for all internal IDs associated with this record.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

### Supported Script Types

The promotion record is scriptable in both client and server SuiteScript.

### Supported Functions

The promotion record is fully scriptable — it can be created, updated, copied, deleted, and searched using SuiteScript.

### Usage Notes

The following table provides usage notes for specific fields on this record.

<table>
<thead>
<tr>
<th>Field Internal ID</th>
<th>Field UI Label</th>
<th>Note</th>
</tr>
</thead>
</table>
| discounttype      | radio buttons  | Valid values in scripts are:  
|                   |                | ✅ percent  
|                   |                | ✅ flat    |
When applying a free gift promotion to a transaction using SuiteScript, the free gift is not automatically added or removed from the order.

The following SuiteScript example shows how to apply the free gift and remove it when the transaction is no longer eligible:

```javascript
// Create Sales Order
var mySalesOrder = record.create({
  type: record.Type.SALES_ORDER,
  isDynamic: true
});
mySalesOrder.setValue({
  fieldId: 'entity',
  value: '41'
});

// Add free gift promotion to the promotion machine
mySalesOrder.selectNewLine({
  sublistId: 'promotions'
});
mySalesOrder.setCurrentSublistValue({
  sublistId: 'promotions',
  fieldId: 'promocode',
  value: '103'
});
mySalesOrder.commitLine({
  sublistId: 'promotions'
});

// Add eligible item with quantity 4 to the item machine
mySalesOrder.setCurrentSublistValue({
  sublistId: 'item',
  fieldId: 'item',
  value: '233'
});
mySalesOrder.setCurrentSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  value: '4'
});
mySalesOrder.commitLine({
  sublistId: 'item'
});

// *********** Manually add free gift and link it to the free gift promotion ***********
mySalesOrder.setCurrentSublistValue({
  sublistId: 'item',
  fieldId: 'item',
  value: '244'
});
mySalesOrder.setCurrentSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  value: '2'
});
mySalesOrder.setCurrentSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  value: '2'
});
```
sublistId: 'item',
fieldId: 'freegiftpromotion',
value: '103'
});
mySalesOrder.commitLine({
  sublistId: 'item'
});

// Check Free Gift Promotion columns on Promotion Machine:
// firstFreeGiftQuantity should say 2
// firstEligibleFreeGifts should say 2 // firstFreeGiftsAdded should say 2
var firstFreeGiftQuantity = mySalesOrder.getSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  line: 2
});
var firstEligibleFreeGifts = mySalesOrder.getSublistValue({
  sublistId: 'promotions',
  fieldId: 'eligiblefreegifts',
  line: 1
});
var firstFreeGiftsAdded = mySalesOrder.getSublistValue({
  sublistId: 'promotions',
  fieldId: 'freegiftsadded',
  line: 1
});

// *********** Reduce eligible item quantity to 3 ***********
mySalesOrder.selectLine({
  sublistId: 'item',
  line: '1'
});
mySalesOrder.setCurrentSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  value: '3'
});
mySalesOrder.commitLine({
  sublistId: 'item'
});

// Verify that the columns have been updated but the user still needs to manually remove 1 Free Gift
// secondFreeGiftQuantity should say 2
// secondEligibleFreeGifts should say 1
// secondFreeGiftsAdded should say 2
var secondFreeGiftQuantity = mySalesOrder.getSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  line: 2
});
var secondEligibleFreeGifts = mySalesOrder.getSublistValue({
  sublistId: 'promotions',
  fieldId: 'eligiblefreegifts',
  line: 1
});
var secondFreeGiftsAdded = mySalesOrder.getSublistValue({
  sublistId: 'promotions',
  fieldId: 'freegiftsadded',
  value: 1
});

// Manually reduce Free Gift quantity
mySalesOrder.selectLine({
  sublistId: 'item',
  line: '2'
});
mySalesOrder.setSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  value: '1'
});
mySalesOrder.commitLine({
  sublistId: 'item'
});

// Verify that the promotions machine columns have been updated again
// Verify that the columns have been updated but the user still needs to manually remove 1 Free Gift
// firstFreeGiftQuantity should say 1
// firstEligibleFreeGifts should say 1
// firstFreeGiftsAdded should say 1
var thirdFreeGiftQuantity= mySalesOrder.getSublistValue({
  sublistId: 'item',
  fieldId: 'quantity',
  line: 2
});
var thirdEligibleFreeGifts = mySalesOrder.getSublistValue({
  sublistId: 'promotions',
  fieldId: 'eligiblefreegifts',
  line: 1
});
var thirdFreeGiftsAdded = mySalesOrder.getSublistValue({
  sublistId: 'promotions',
  fieldId: 'freegiftsadded',
  line: 1
});

Working with the Discount Amount Search Column

When you use the Discount Amount search column, instead of using the discountamount internal ID, use discountrate.

Website

Note: This topic applies to all versions of SuiteScript.

The following website records are scriptable in SuiteScript:

- CMS Content
CMS Content

Note: This topic applies to all versions of SuiteScript.

CMS content enables you to create things such as text and images for your website. For help working with this record in the UI, see the help topic Site Management Tools.

The internal ID for this record is cmscontent.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The CMS content record is scriptable in server SuiteScript only.

Supported Functions

The CMS content record is partially scriptable. It can be created, updated, and searched using SuiteScript. It cannot be copied, attached, or transformed.

Usage Notes

You must enable the Site Management Tools feature on the Web Presence subtab at Setup > Company > Enable Features to be able to script with this record.

Script Sample

The following sample code snippet creates a script that will search for a CMS content record.

```javascript
function searchCmsContentRecords(siteId){
  var mySearch = search.create({
    type: search.Type.CMS_CONTENT,
    filters: [{
      name: 'site',
      operator: search.Operator.IS,
      values: 'siteId'
    }]
  });
  return mySearch.run();
}
```
CMS Content Type

**Note:** This topic applies to all versions of SuiteScript.

CMS page enables you to create different content types for your website. For more information, see the help topic Site Management Tools.

For help working with this record in the UI, see CMS Content.

The internal ID for this record is **cmspage**.

See the [SuiteScript Records Browser](#) for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic [Working with the SuiteScript Records Browser](#) in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

### Supported Script Types

The CMS content type record is scriptable in server SuiteScript only.

### Supported Functions

The CMS content type record is partially scriptable. It can be created, updated, and searched using SuiteScript. It cannot be copied, attached, or transformed.
Usage Notes

You must enable the Site Management Tools feature on the Web Presence subtab at Setup > Company > Enable Features to be able to script with this record.

Script Samples

The following sample code snippets show how to create and search for CMS content type records. All CMS content type records are associated with a custom record. For more information, see the help topics Custom Records and Creating Custom Record Types

Create CMS Content Type Record

The following sample code snippet creates a basic CMS content type record and associates it with the specified custom record.

```javascript
var cmsContentTypeRecord = record.create({
  type: record.Type.CMS_CONTENT_TYPE
});

// Set field values

// Name
var nameField = record.createField(record.Type.CREATABLE_TEXT);
nameField.setValues({
  fieldId: 'name',
  value: 'cms content type name'
});

// Label
var labelField = record.createField(record.Type.CREATABLE_TEXT);
labelField.setValues({
  fieldId: 'label',
  value: 'content type label'
});

// Description
var descriptionField = record.createField(record.Type.CREATABLE_TEXT);
descriptionField.setValues({
  fieldId: 'description',
  value: 'My cms content type description'
});

// Custom Record ID
var customRecordIdField = record.createField(record.Type.INTEGER);
customRecordIdField.setValues({
  fieldId: 'customrecordid',
  value: 123                  // example value
});

var contentTypeID = cmsContentTypeRecord.save();
```

Search CMS Content Type Records

The following sample code snippet searches for all CMS content type records for a given SCA site. It returns the name, site, pagetype, and url fields as search result columns.

```javascript
function searchCmsContentTypeRecords(){
  var searchColumns = search.create({
    type: search.Type.CMS_CONTENT_TYPE,
    columns: 
      [{
        name: 'name'
      },
      { name: 'label'
      },
      { name: 'customrecordid'
      }]
  });

  searchColumns.run().each(function(result{...
```
CMS Page

Note: This topic applies to all versions of SuiteScript.

CMS page enables you to create different pages for hosting content on your website. For more information, see the help topic Pages in SMT.

For help working with this record in the UI, see CMS Page.

The internal ID for this record is `cmspage`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

Note: For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- N/record Module

Supported Script Types

The CMS page record is scriptable in server SuiteScript only.

Supported Functions

The CMS page record is partially scriptable. It can be created, updated, and searched using SuiteScript. It cannot be copied, attached, or transformed.

Usage Notes

You must enable the Site Management Tools feature on the Web Presence subtab at Setup > Company > Enable Features to be able to script with this record.

Script Samples

Create CMS Page Record
The following sample code snippet creates a basic landing page record (pagetype = 1) for the specified SCA site.

```javascript
var recordObj = record.create({
    type: record.Type.CMS_PAGE,
    isDynamic: false
});
recordObj.setValue({
    fieldId: 'name',
    value: 'Cms Page Name'
});
recordObj.setValue({
    fieldId: 'site',
    value: 1
});
recordObj.setValue({
    fieldId: 'pagetype',
    value: 1
});
recordObj.setValue({
    fieldId: 'cmspagetype',
    value: 1
});
recordObj.setValue({
    fieldId: 'url',
    value: 'PageUrl'
});
return = recordObj.save({
    enableSourcing: false,
    ignoreMandatoryFields: false
});
```

**Search CMS Page Records**

The following sample code snippet searches for all CMS page records for a given SCA site. It returns the name, site, pagetype, and url fields as search result columns.

```javascript
function searchCmsPageRecords(siteId){
    var searchColumns = search.create({
        type: search.Type.CMS_PAGE,
        columns: [{
            name: 'name'
        }, {
            name: 'site'
        }, {
            name: 'pagetype'
        }, {
            name: 'url'
        }],
        filters: [{
            name: 'site',
            operator: search.Operator.IS,
            values: 1  // example siteld
        }]
    });
    searchColumns.run().each(function(result){
        // process result
    });
}
```
var name = result.getValue({
    name: 'name'
});
var site = result.getValue({
    name: 'site'
});
var pageType = result.getValue({
    name: 'pagetype'
});
var url = result.getValue({
    name: 'url'
});
return true;
}

Commerce Category

**Note:** This topic applies to all versions of SuiteScript.

Commerce Categories enable you to create a hierarchical structure of product categories, subcategories, and products.

For help working with this record in the UI, see the help topic Commerce Categories.

In the UI, you can go to Lists > Web Site > Commerce Categories to access this record.

The internal ID for this record is `commercecategory`.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
- SuiteScript 2.0 Custom Forms
- `N/record` Module

**Supported Script Types**

The commerce category record is scriptable in server SuiteScript only.

**Supported Functions**

The commerce category record is partially scriptable. It can be created, updated, and searched using SuiteScript. It cannot be copied, attached, or transformed.

**Usage Notes**

You must enable the Commerce Category feature on the Web Presence subtab at Setup > Company > Enable Features to be able to script with this record.
Script Sample

The following sample code snippet searches for a commerce category record with a fullurl of "/cat-0" and siteid of 2. It returns fullurl, addtohead, and subcataddtoheadoverride as search result columns.

```javascript
var fullurl = '/cat-0';
var siteid = 2;

var mySearch = search.create({
    type: search.Type.COMMERCE_CATEGORY,
    filters: [{
        name: 'fullurl',
        operator: search.Operator.IS,
        values: fullurl
    }],
    columns: [{
        name: 'fullurl'
    }, {
        name: 'addtohead'
    }, {
        name: 'subcataddtoheadoverride'
    }]
});
mySearch.run().each(function(result){
    var name = result.getValue({
        name: 'fullurl'
    });
    var site = result.getValue({
        name: 'addtohead'
    });
    var pageType = result.getValue({
        name: 'subcataddtoheadoverride'
    });
    return true;
});
```

Website Setup

**Note:** This topic applies to all versions of SuiteScript.

The internal ID for this record is `website`. In the UI, you can find this record by going to Setup > Web Site > Set Up Web Site.

For help working with this record in the UI, see the help topic Web Site Preferences.

See the SuiteScript Records Browser for all internal IDs associated with this record.

**Note:** For information on using the SuiteScript Records Browser, see the help topic Working with the SuiteScript Records Browser in the NetSuite Help Center.

For information about scripting with this record in SuiteScript, see the following:

- SuiteScript 2.0 Scripting Records and Subrecords
Supported Script Types

The web site setup record is scriptable in server SuiteScript only. The web site record is not supported in `beforeLoad` user event scripts.

Supported Functions

The web site setup record is fully scriptable. It can be created, copied, updated, deleted, and searched using SuiteScript.

Usage Notes

Developers can use SuiteScript with the web site setup record whether they are working in accounts using Site Builder or SuiteCommerce Advanced.

Creating a Web Site Setup Record in SuiteScript

To create a Web Site Setup record in a Site Builder site, developers can write code using one of two approaches:

The first approach is:

```
var recWebSite = record.create(
    type: record.Type.WEBSITE
);
```

The second approach is:

```
var initvalues = new Array();
initvalues.sitetype = 'STANDARD';
initvalues.websitescope = 'SUITE_COMMERCE';
var recWebSite = record.create(
    type: record.Type.WEBSITE, 
    defaultValues: initvalues
);
```

In the first approach, the `initvalues` are defaulted to STANDARD. However, it is recommended that developers use the second approach because there are two website types. To clearly distinguish between the two types in your code, the second approach is cleaner.

To create a Web Site Setup record in a SuiteCommerce Advanced site, the code must look like the following:

```
var initvalues = new Array();
initvalues.sitetype = 'ADVANCED';
initvalues.websitescope = 'SUITE_COMMERCE';
var recWebSite = record.create(
    type: record.Type.WEBSITE, 
    defaultValues: initvalues
);
```
Websitescope field for ADVANCED sites is read-only by default. This means that you cannot modify websitescope for an existing ADVANCED site. To ensure that your site has the correct websitescope, you need to specify the required value in defaultValues when creating the site. Below you can find a list of all the possible ADVANCED websitescope values for each of the different ADVANCED site types:

<table>
<thead>
<tr>
<th>Site Type</th>
<th>Websitescope value</th>
</tr>
</thead>
<tbody>
<tr>
<td>SuiteCommerce</td>
<td>SUITE_COMMERCE</td>
</tr>
<tr>
<td>SuiteCommerce Advanced</td>
<td>SUITE_COMMERCE_ADVANCED</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> This is the default value.</td>
</tr>
<tr>
<td>SuiteCommerce InStore</td>
<td>SUITE_COMMERCE_IN_STORE</td>
</tr>
<tr>
<td>SuiteCommerce MyAccount</td>
<td>SUITE_COMMERCE_MY_ACCOUNT</td>
</tr>
</tbody>
</table>

**Setting Values for Web Site Setup Dropdown Fields**

The following table lists the dropdown fields on the Web Site Setup record. When writing SuiteScript, if you are setting the value of a dropdown field, use the IDs listed in the column called Internal IDs for Dropdown Values.

<table>
<thead>
<tr>
<th>Tab</th>
<th>Dropdown Field</th>
<th>UI Labels for Dropdown Values</th>
<th>Internal IDs for Dropdown Values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup tab</strong></td>
<td>Web Site Scope</td>
<td>Full Web Store</td>
<td>FULL_WEB_STORE</td>
</tr>
<tr>
<td></td>
<td>(websitescope)</td>
<td>Information And Catalog, With Pricing</td>
<td>INFO_CATALOG_PRICING</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Information And Catalog</td>
<td>INFO_CATALOG</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Information Only</td>
<td>INFO_ONLY</td>
</tr>
<tr>
<td></td>
<td>Default Customer Category</td>
<td>Corporate</td>
<td>CORPORATE</td>
</tr>
<tr>
<td></td>
<td>(defaultcustomercategory)</td>
<td>Individual</td>
<td>INDIVIDUAL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Employee</td>
<td>EMPLOYEE</td>
</tr>
<tr>
<td><strong>Appearance tab</strong></td>
<td>Web Site Logo Alignment</td>
<td>Align Left</td>
<td>LEFT</td>
</tr>
<tr>
<td></td>
<td>(websitelogoalign)</td>
<td>Align Right</td>
<td>RIGHT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Align Center</td>
<td>CENTER</td>
</tr>
<tr>
<td></td>
<td>Page Alignment</td>
<td>Align Left</td>
<td>LEFT</td>
</tr>
<tr>
<td></td>
<td>(pagealign)</td>
<td>Align Right</td>
<td>RIGHT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Align Center</td>
<td>CENTER</td>
</tr>
<tr>
<td></td>
<td>Display Order of Cart Items</td>
<td>Most Recently Added First</td>
<td>RECENT_FIRST</td>
</tr>
<tr>
<td>Tab</td>
<td>Dropdown Field</td>
<td>UI Labels for Dropdown Values</td>
<td>Internal IDs for Dropdown Values</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>Upsell tab</td>
<td>Items to Upsell (upsellitems)</td>
<td>■ Most Recently Added Last</td>
<td>■ RECENT_LAST</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Show Related Items First and Upsell Items Next</td>
<td>■ RELATED_FIRST_UPSELL_NEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Show Upsell Items First and Related Items Next</td>
<td>■ UPSELL_FIRST_RELATED_NEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Show Only Related Items</td>
<td>■ ONLY_RELATED_ITEMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Show Only Upsell Items</td>
<td>■ ONLY_UPSELL_ITEMS</td>
</tr>
<tr>
<td></td>
<td>Items to Upsell in Cart (cartupsellitems)</td>
<td>■ Show Related Items First and Upsell Items Next</td>
<td>■ RELATED_FIRST_UPSELL_NEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Show Upsell Items First and Related Items Next</td>
<td>■ UPSELL_FIRST_RELATED_NEXT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Show Only Related Items</td>
<td>■ ONLY_RELATED_ITEMS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Show Only Upsell Items</td>
<td>■ ONLY_UPSELL_ITEMS</td>
</tr>
<tr>
<td>Legacy tab</td>
<td>Site Tab Alignment (sitetabalignment)</td>
<td>■ Align Left</td>
<td>■ LEFT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Align Right</td>
<td>■ RIGHT</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Align Center</td>
<td>■ CENTER</td>
</tr>
<tr>
<td>Shopping tab</td>
<td>Sales Order Type (salesordertype)</td>
<td>■ Per Customer Basis</td>
<td>■ PER_CUSTOMER</td>
</tr>
</tbody>
</table>

### Shopping Cart

**Note:** This topic applies to all versions of SuiteScript.

For help working with this record in the UI, see the help topic *Shopping Cart and Checkout Set Up*.

The internal ID for this record is `shoppingcart`. You can use the UI to create saved searches with data from the shopping cart. Go to Lists > Search > Saved Searches > New > Shopping cart.

For information about scripting with this record in SuiteScript, see the following:

- [SuiteScript 2.0 Scripting Records and Subrecords](#)
- [SuiteScript 2.0 Custom Forms](#)
- [N/record Module](#)

### Supported Script Types

The shopping cart record is scriptable in server SuiteScript only.

### Supported Functions

The shopping cart record is not fully scriptable. Only search is permitted.
Usage Notes

You can use SuiteScript to create sophisticated marketing campaigns based upon the data linked to the shopping cart. The following data points are exposed through SuiteScript: customer ID, date created, date edited, item IDs for cart items, and website ID.

Note: Because shopping carts can be created by anonymous shoppers, in some search results the customer ID may be null.

Script Samples

The following example uses the `search.create(options)` method to get shopping cart information.

To get carts from a certain website (id:1) which contains a particular item (item id:352), use the following code sample:

```javascript
var mySearch = search.create(
    { 
      type: search.Type.SHOPPING_CART, 
      filters: [{ 
        name: 'websiteid', 
        operator: search.Operator.EQUALTO, 
        values: '1' 
      }, { 
        name: 'itemid', 
        operator: search.Operator.EQUALTO, 
        values: '352' 
      }], 
      columns: [{
        name: 'itemid'
      }, {
        name: 'itemqty'
      }, { 
        name: 'websiteid'
      }, { 
        name: 'customerid'
      }]
    });
mySearch.run().each(function(result){
  var name = result.getValue({
    name: 'fullurl'
  });
  var site = result.getValue({
    name: 'addtohead'
  });
  var pageType = result.getValue({
    name: 'subcataddtoheadoverride'
  });
  return true;
});
```

You can also use `n1apiLoadSearch` to programmatically execute a previously created shopping cart saved search. Or, you can use methods from the SuiteScript 2.0 N/search module.