and 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.
# Table of Contents

SuiteScript 1.0 to SuiteScript 2.0 API Map ................................................................. 1  
SuiteScript 1.0 to SuiteScript 2.0 API Map – Functions (nlapi) ................................. 1  
SuiteScript 1.0 to SuiteScript 2.0 API Map – Objects (nlobj) ........................................ 10  
SuiteScript 2.0 Global Objects ................................................................................... 35  

define Object ............................................................................................................ 35  
define(moduleObject) ................................................................................................... 36  
define(id, [dependencies] moduleObject) ................................................................. 37  
require Function ......................................................................................................... 40  
require([dependencies,] callback) ............................................................................. 40  
require Configuration ................................................................................................. 42  
log Object .................................................................................................................. 43  
util Object .................................................................................................................. 43  
toString() .................................................................................................................. 44  
JSON object .............................................................................................................. 44  
JSON.parse(text) ......................................................................................................... 45  
JSON.stringify(obj) ................................................................................................. 45  
Promise Object .......................................................................................................... 46  

SuiteScript 2.0 Modules ........................................................................................... 50  

N/action Module ........................................................................................................ 52  
action.Action ............................................................................................................. 57  
action.execute(options) ............................................................................................. 67  
action.execute.promise(options) .............................................................................. 68  
action.executeBulk(options) (Beta) ......................................................................... 70  
action.find(options) .................................................................................................. 71  
action.find.promise(options) ................................................................................... 73  
action.get(options) ................................................................................................... 74  
action.get.promise(options) .................................................................................... 75  
N/auth Module .......................................................................................................... 76  
auth.changeEmail(options) ....................................................................................... 77  
auth.changePassword(options) .................................................................................. 78  
N/cache Module ........................................................................................................ 79  
cache.Cache ............................................................................................................. 82  
cache.getCache(options) .......................................................................................... 88  
cache.Scope ............................................................................................................ 88  
N/certificateControl Module .................................................................................... 89  
certificateControl.findCertificates(options) ............................................................ 90  
certificateControl.Type .......................................................................................... 91  
N/config Module ....................................................................................................... 92  
config.load(options) ............................................................................................... 93  
config.Type .............................................................................................................. 94  
N/crypto Module ..................................................................................................... 96  
crypto.Cipher .......................................................................................................... 100  
crypto.CipherPayload .............................................................................................. 102  
crypto.Decipher ...................................................................................................... 104  
crypto.Hash ............................................................................................................ 106  
crypto.Hmac ........................................................................................................... 108  
crypto.SecretKey ................................................................................................. 110  
crypto.createCipher(options) ............................................................................... 112  
crypto.createDecipher(options) ............................................................................ 113  
crypto.createHash(options) .................................................................................. 114  
crypto.createHmac(options) .................................................................................. 114  
crypto.createSecretKey(options) .......................................................................... 115  
crypto.EncryptionAlg ............................................................................................ 116
<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>render</td>
<td>673</td>
</tr>
<tr>
<td>redirect</td>
<td>665</td>
</tr>
<tr>
<td>record</td>
<td>524</td>
</tr>
<tr>
<td>query</td>
<td>423</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method/API</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>render.bom(options)</td>
<td>687</td>
</tr>
<tr>
<td>render.TemplateRenderer</td>
<td>679</td>
</tr>
<tr>
<td>render.EmailMergeResult</td>
<td>677</td>
</tr>
<tr>
<td>redirect.toTaskLink(options)</td>
<td>672</td>
</tr>
<tr>
<td>redirect.toSearchResult(options)</td>
<td>671</td>
</tr>
<tr>
<td>redirect.toSearch(options)</td>
<td>670</td>
</tr>
<tr>
<td>redirect.toSavedSearch(options)</td>
<td>668</td>
</tr>
<tr>
<td>record.transform.promise(options)</td>
<td>660</td>
</tr>
<tr>
<td>record.submitFields.promise(options)</td>
<td>655</td>
</tr>
<tr>
<td>record.submitFields(options)</td>
<td>652</td>
</tr>
<tr>
<td>record.load(options)</td>
<td>648</td>
</tr>
<tr>
<td>record.delete.promise(options)</td>
<td>643</td>
</tr>
<tr>
<td>record.delete(options)</td>
<td>645</td>
</tr>
<tr>
<td>record.attach.promise(options)</td>
<td>630</td>
</tr>
<tr>
<td>record.attach(options)</td>
<td>634</td>
</tr>
<tr>
<td>record.copypromise(options)</td>
<td>632</td>
</tr>
<tr>
<td>record.copy(options)</td>
<td>634</td>
</tr>
<tr>
<td>record.create.promise(options)</td>
<td>635</td>
</tr>
<tr>
<td>record.create(options)</td>
<td>638</td>
</tr>
<tr>
<td>query.load(options)</td>
<td>486</td>
</tr>
<tr>
<td>query.delete(options)</td>
<td>485</td>
</tr>
<tr>
<td>query.createDateRange(options)</td>
<td>484</td>
</tr>
<tr>
<td>query.delete(options)</td>
<td>486</td>
</tr>
<tr>
<td>query.Aggregate</td>
<td>488</td>
</tr>
<tr>
<td>query.DateRangeId</td>
<td>489</td>
</tr>
<tr>
<td>query.Operator</td>
<td>491</td>
</tr>
<tr>
<td>query.RelativeDateRange</td>
<td>493</td>
</tr>
<tr>
<td>query.ReturnType</td>
<td>498</td>
</tr>
<tr>
<td>query.SortLocale</td>
<td>500</td>
</tr>
<tr>
<td>query.Type</td>
<td>506</td>
</tr>
<tr>
<td>N/render Module</td>
<td>673</td>
</tr>
<tr>
<td>render.EmailMergeResult</td>
<td>677</td>
</tr>
<tr>
<td>render.TemplateRenderer</td>
<td>679</td>
</tr>
<tr>
<td>render.bom(options)</td>
<td>687</td>
</tr>
</tbody>
</table>
N/sftp Module ................................................................. 810
Setting up an SFTP Transfer .............................................. 813
SFTP Authentication .......................................................... 814
Supported Cipher Suites and Host Key Types ....................... 816
Supported SuiteScript File Types ....................................... 816
sftp.Connection ................................................................ 818
sftp.createConnection(options) ........................................ 822
N/sso Module ................................................................... 825
sso.generateSuiteSignOnToken(options) ............................ 825
N/task Module .................................................................. 828
task.ScheduledScriptTask .................................................. 845
task.ScheduledScriptTaskStatus ........................................ 848
task.MapReduceScriptTask ............................................... 850
task.MapReduceScriptTaskStatus ...................................... 854
task.CsvImportTask ......................................................... 865
task.CsvImportTaskStatus ............................................... 869
task.EntityDeduplicationTask ........................................... 871
task.EntityDeduplicationTaskStatus .................................. 875
task.SearchTask ............................................................. 877
task.SearchTaskStatus ..................................................... 884
task.WorkflowTriggerTask ................................................. 887
task.WorkflowTriggerTaskStatus ....................................... 890
task.RecordActionTask (Beta) ............................................. 892
RecordActionTask.submit() ................................................. 892
RecordActionTask.toString() ............................................. 893
RecordActionTask.toJSON() ................................................. 894
RecordActionTask.paramCallback() ................................. 895
RecordActionTask.recordType ............................................. 895
RecordActionTask.action .................................................. 896
RecordActionTask.params ............................................... 897
RecordActionTask.condition .............................................. 898
RecordActionTaskTaskStatus (Beta) .................................... 899
RecordActionTaskStatus.toString() .................................. 901
RecordActionTaskStatus.toJSON() .................................... 901
RecordActionTaskStatus.status ......................................... 902
RecordActionTaskStatus.results ....................................... 903
RecordActionTaskStatus.errors ....................................... 904
RecordActionTaskStatus.complete .................................... 904
RecordActionTaskStatus.succeeded ................................... 905
RecordActionTaskStatus.failed ....................................... 906
RecordActionTaskStatus.pending ...................................... 907
task.create(options) .......................................................... 908
task.checkStatus(options) .................................................. 912
task.TaskType .............................................................. 913
task.TaskStatus ............................................................ 914
task.MasterSelectionMode ................................................ 915
task.DedupeMode .......................................................... 916
task.DedupeEntityEntityType .......................................... 916
task.MapReduceStage ...................................................... 917
taskActionCodeCondition (Beta) ........................................ 918
N/transaction Module ....................................................... 919
transaction.void(options) .................................................. 921
transaction.void.promise(options) ................................. 922
transaction.Type .......................................................... 923
N/translation Module ........................................................ 925
util.each(iterable, callback) ................................................................. 1102
util.extend(receiver, contributor) ........................................................ 1103
N/workflow Module .............................................................................. 1104
    workflow.initiate(options) ............................................................. 1106
    workflow.trigger(options) ............................................................. 1107
N/xml Module ...................................................................................... 1108
    xml.Parser .................................................................................. 1117
    xml.XPath .................................................................................... 1119
    xml.Node .................................................................................... 1120
    xml.Document ............................................................................. 1138
    xml.Element .............................................................................. 1154
    xml.Attr ..................................................................................... 1167
    xml.escape(options) .................................................................. 1169
    xml.validate(options) ................................................................ 1170
    xml.NodeType .............................................................................. 1171
SuiteScript 1.0 to SuiteScript 2.0 API Map

**Important:** These topics are a work in progress. Some items are currently missing or do not have content. Additional updates are forthcoming.

These topics map SuiteScript 1.0 APIs to their corresponding SuiteScript 2.0 APIs. Keep the following in mind when using these mappings:

- Some SuiteScript 1.0 APIs do not have a SuiteScript 2.0 equivalent.
- There is not always a one to one mapping between SuiteScript 1.0 and SuiteScript 2.0. Each SuiteScript 1.0 API is listed only one time, but it may map to several SuiteScript 2.0 APIs.
- These mappings **do not include** SuiteScript 1.0 deprecated APIs.
- These mappings **do not include** new SuiteScript 2.0 functionality. To find new SuiteScript 2.0 functionality, go to SuiteScript 2.0 Modules. The table includes a description of, and link to, each module.

**Important:** If you are using SuiteScript 1.0 for your scripts, consider converting these scripts to SuiteScript 2.0. Use SuiteScript 2.0 to take advantage of new features, APIs, and functionality enhancements. For more information, see the help topic SuiteScript 2.0 Advantages.

These topics group SuiteScript 1.0 APIs into functions (prefixed with “nlapi”) and objects (prefixed with “nlobj”). All functions are listed alphabetically in one table. Whereas objects and their members are grouped alphabetically by object name. Each object has its own table containing all object members.

- SuiteScript 1.0 to SuiteScript 2.0 API Map – Functions (nlapi)
- SuiteScript 1.0 to SuiteScript 2.0 API Map – Objects (nlobj)

### SuiteScript 1.0 to SuiteScript 2.0 API Map – Functions (nlapi)

This topic maps SuiteScript 1.0 Functions (prefixed with “nlapi”) to their corresponding SuiteScript 2.0 APIs. All functions are listed alphabetically in one table.

**Note:** NetSuite does not support calling SuiteScript 1.0 APIs from SuiteScript 2.0 scripts.

**Note:** To view a mapping of SuiteScript 1.0 Objects (prefixed with “nlobj”) to their corresponding SuiteScript 2.0 APIs, see SuiteScript 1.0 to SuiteScript 2.0 API Map – Objects (nlobj).

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlapiAddDays(d, days)</td>
<td>See Notes</td>
<td>See Notes</td>
<td>This API does not have a SuiteScript 2.0 equivalent. Use the following JavaScript to add or subtract days from a Date object: dateObj.setDate(dateObj.getDate() + or – days) For example: var tomorrow = new Date(); tomorrow.setDate(tomorrow.getDate() + 1);</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-----------------------</td>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **nlapiAddMonths(d, months)**                           | See Notes             | See Notes              | This API does not have a SuiteScript 2.0 equivalent. **Note:** SuiteScript 2.0 is also compatible with third party JavaScript APIs that provide this functionality (for example, Moment.js). For information on using third party APIs with SuiteScript 2.0, see the help topic SuiteScript 2.0 Custom Modules. Use the following JavaScript to add or subtract months from a Date object: `dateObj.setMonth(dateObj.getMonth() + or – months)` For example: ```
``` js
```javascript
var today = new Date();
var oneMonthAgo =
today.setMonth(today.getMonth() - 1);
``` |
<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlapiCreateCurrentLineItemSubrecord(fldname)</td>
<td>Record CurrentRecord Subrecord fldname</td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlapiCreateEmailMerger(templated)</td>
<td>render.mergeEmail(options)</td>
<td>N/render Module</td>
<td>For a script sample, see N/render Module Script Sample.</td>
</tr>
<tr>
<td>nlapiCreateError(code, details, suppressNotification)</td>
<td>error.create(options)</td>
<td>N/error Module</td>
<td>For a script sample, see N/error Module Script Samples.</td>
</tr>
<tr>
<td>nlapiCreateFile(name, type, contents)</td>
<td>file.create(options)</td>
<td>N/file Module</td>
<td>For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td>nlapiCreateForm(title, hideNavbar)</td>
<td>serverWidget.create(options)</td>
<td>N/ui/serverWidget Module</td>
<td>For a script sample, see N/ui/serverWidget Module Script Samples.</td>
</tr>
<tr>
<td>nlapiCreateList(title, hideNavbar)</td>
<td>serverWidget.create(options)</td>
<td>N/ui/serverWidget Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlapiCreateRecord(type, initializeValues)</td>
<td>record.create(options)</td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlapiCreateSearch(type, filters, columns)</td>
<td>search.create(options)</td>
<td>N/search Module</td>
<td>For a script sample, see N/search Module Script Samples.</td>
</tr>
<tr>
<td>nlapiCreateSubrecord(fldname)</td>
<td>Record.getSubrecord(options)</td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlapiDateToString(d, format)</td>
<td>format.format(options)</td>
<td>N/format Module</td>
<td>For a script sample, see N/format Module Script Samples.</td>
</tr>
<tr>
<td>nlapiDeleteFile(id)</td>
<td>file.delete(options)</td>
<td>N/file Module</td>
<td></td>
</tr>
<tr>
<td>nlapiDeleteRecord(type, id, initializeValues)</td>
<td>record.delete(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlapiDetachRecord(type, id, type2, id2, attributes)</td>
<td>record.detach(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlapiDisableField(fldnam, val)</td>
<td>Field.isDisabled</td>
<td>N/current Record Module</td>
<td>Note that isDisabled is a property.</td>
</tr>
<tr>
<td>nlapiDisableCurrentLineItemField(type, fldnam, val)</td>
<td>Field.isDisabled</td>
<td>N/current Record Module</td>
<td>Note that isDisabled is a property.</td>
</tr>
<tr>
<td>nlapiEditCurrentLineItemSubrecord(fldname)</td>
<td>Record.getSubrecord(options)</td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlapiEditSubrecord(fldname)</td>
<td>Record.getSubrecord(options)</td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlapiEncrypt(s, algorithm, key)</td>
<td>See Notes</td>
<td>See Notes</td>
<td>For SuiteScript 2.0 encryption, hashing, and HMAC functionality, see the N/crypto Module. For SuiteScript 2.0 encoding functionality, see the N/encode Module.</td>
</tr>
<tr>
<td>nlapiEscapeXML(text)</td>
<td>xml.escape(options)</td>
<td>N/xml Module</td>
<td></td>
</tr>
</tbody>
</table>

**SuiteScript 2.0 API Reference**

**Oracle NetSuite**
<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlapiExchangeRate(sourceCurrency, targetCurrency, effectiveDate)</td>
<td>currency.exchangeRate(options)</td>
<td>-</td>
<td>For a script sample, see N/currency Module Script Sample.</td>
</tr>
<tr>
<td>nlapiFindLineItemMatrixValue(type, fldnam, val, column)</td>
<td>Record.findMatrixSublistLineWithValue(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiFindLineItemValue(type, fldnam, val)</td>
<td>CurrentRecord.findSublistLineWithValue(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiFormatCurrency(str)</td>
<td>format.format(options)</td>
<td>N/format Module</td>
<td>Note that SuiteScript 2.0 currency formatting is handled by the N/format module and not the N/currency module. For a script sample, see N/format Module Script Samples.</td>
</tr>
<tr>
<td>nlapiGetContext()</td>
<td>runtime.getCurrentScript()</td>
<td>N/runtime Module</td>
<td>For a script sample, see N/runtime Module Script Sample.</td>
</tr>
<tr>
<td>nlapiGetCurrentLineItemDateValue(type, fieldId, timeZone)</td>
<td>See Notes</td>
<td>N/format Module</td>
<td>Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
<tr>
<td>nlapiGetCurrentLineItemIndex(type)</td>
<td>CurrentRecord.getCurrentSublistIndex(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetCurrentLineItemMatrixValue(type, fldnam, column)</td>
<td>CurrentRecord.getCurrentMatrixSublistValue(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetCurrentLineItemText(fldnam)</td>
<td>CurrentRecord.getCurrentSublistText(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetCurrentLineItemValue(fldnam)</td>
<td>CurrentRecord.getCurrentSublistValue(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetCurrentLineItemValues(fldnam)</td>
<td>CurrentRecord.getCurrentSublistValue(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetDateTimeValue(fieldId, timeZone)</td>
<td>See Notes</td>
<td>N/format Module</td>
<td>Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
<tr>
<td>nlapiGetDepartment()</td>
<td>User.department</td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetField(fldnam)</td>
<td>Record.getField(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetFieldText(fldnam)</td>
<td>Record.getText(options)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlapiGetFieldTexts(fldnam)</td>
<td>Record.getText(Record, Module)</td>
<td>CurrentRecord.getText(Record, Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetFieldValue(fldnam)</td>
<td>Record.getValue(Module)</td>
<td>CurrentRecord.getValue(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetFieldValues(fldnam)</td>
<td>Record.getValue(Module)</td>
<td>CurrentRecord.getValue(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetJobManager(jobType)</td>
<td>task.create(task Module)</td>
<td>For a script sample, see N/task Module Script Samples.</td>
<td></td>
</tr>
<tr>
<td>nlapiGetLineItemCount(type)</td>
<td>Record.getLineCount(Module)</td>
<td>CurrentRecord.getLineCount(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetLineItemField(type, fldnam, linenum)</td>
<td>Record.getSublistField(Module)</td>
<td>CurrentRecord.getSublistField(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetLineItemMatrixField(type, fldnam, linenum, column)</td>
<td>Record.getMatrixSublistField(Module)</td>
<td>CurrentRecord.getMatrixSublistField(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetLineItemMatrixValue(type, fldnam, linenum, column)</td>
<td>Record.getMatrixSublistValue(Module)</td>
<td>CurrentRecord.getMatrixSublistValue(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetLineItemText(type, fldnam, linenum)</td>
<td>Record.getMatrixSublistText(Module)</td>
<td>CurrentRecord.getMatrixSublistText(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetLineItemValue(type, fldnam, linenum)</td>
<td>Record.getMatrixSublistValue(Module)</td>
<td>CurrentRecord.getMatrixSublistValue(Module)</td>
<td>Method returns an array for multi-select fields. SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>nlapiGetLocation()</td>
<td>User.location</td>
<td>N/runtime Module</td>
<td>Note that location is a property.</td>
</tr>
<tr>
<td>nlapiGetLogin()</td>
<td>auth.changeEmail(auth Module)</td>
<td>auth.changePassword(auth Module)</td>
<td>For a script sample, see N/auth Module Script Sample.</td>
</tr>
<tr>
<td>nlapiGetMatrixCount(type, fldnam)</td>
<td>Record.getMatrixHeaderCount(Module)</td>
<td>CurrentRecord.getMatrixHeaderCount(Module)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlapiGetMatrixField(type, fldnam, column)</td>
<td>Record.##getMatrixField(##options)</td>
<td>SuiteScript 1.0 Module</td>
<td>CurrentRecord.##getMatrixField(##options)</td>
</tr>
<tr>
<td>nlapiGetMatrixValue(type, fldnam, column)</td>
<td>N/record Module</td>
<td>Record.##getMatrixValue(##options)</td>
<td>CurrentRecord.##getMatrixValue(##options)</td>
</tr>
<tr>
<td>nlapiGetNewRecord()</td>
<td>See Notes</td>
<td>See Notes</td>
<td>To mimic this functionality in SuiteScript 2.0, use the following code in a beforeLoad(scriptContext), beforeSubmit(scriptContext), or afterSubmit(scriptContext) user event script.</td>
</tr>
<tr>
<td>nlapiGetOldRecord()</td>
<td>See Notes</td>
<td>See Notes</td>
<td>To mimic this functionality in SuiteScript 2.0, use the following code in a beforeSubmit(scriptContext) or afterSubmit(scriptContext) user event script.</td>
</tr>
<tr>
<td>nlapiGetRecordId()</td>
<td>Record.id</td>
<td>N/record Module</td>
<td>CurrentRecord.id</td>
</tr>
<tr>
<td>nlapiGetRecordType()</td>
<td>Record.type</td>
<td>N/record Module</td>
<td>CurrentRecord.##type</td>
</tr>
<tr>
<td>nlapiGetRole()</td>
<td>User.role</td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetSubsidiary()</td>
<td>User.subsidiary</td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td>nlapiGetUser()</td>
<td>runtime.##getCurrentUser()</td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td>nlapiInitiateWorkflow(recordtype, id, workflowid, initialvalues)</td>
<td>workflow.##initiate(##options)</td>
<td>N/workflow Module</td>
<td>For a script sample, see N/workflow Module Script Sample.</td>
</tr>
<tr>
<td>nlapiInsertLineItem(type, line)</td>
<td>Record.##insertLine(##options)</td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlapiInsertLineItem(fldnam, value, text, selected)</td>
<td>CurrentRecord.insertLineItem(options)</td>
<td>N/record Module</td>
<td>Note that isChanged is a property</td>
</tr>
<tr>
<td>nlapiInsertSelectOption(fldnam, field, insertSelectOption(options))</td>
<td>Field.insertSelectOption(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------</td>
</tr>
<tr>
<td><code>nlapiRequestURL</code></td>
<td><code>http</code></td>
<td><code>http</code></td>
<td>SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td><code>nlapiResubmitRequest</code></td>
<td><code>http</code></td>
<td><code>http</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiRequestURLWithCredentials</code></td>
<td><code>https</code></td>
<td><code>https</code></td>
<td>For a script sample, see <code>N/portlet Module</code>.</td>
</tr>
<tr>
<td><code>nlapiResizePortlet</code></td>
<td><code>portlet.resize</code></td>
<td><code>N/portlet Module</code></td>
<td>For a script sample, see <code>N/url Module Script Samples.</code></td>
</tr>
<tr>
<td><code>nlapiResolveURL</code></td>
<td><code>url.resolveRecord</code></td>
<td><code>url Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiResolveURLWithCredentials</code></td>
<td><code>url.resolveScript</code></td>
<td><code>url Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiResolveTaskLink</code></td>
<td><code>url.resolveTaskLink</code></td>
<td><code>url Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSearchDuplicate</code></td>
<td><code>search.duplicates</code></td>
<td><code>N/search Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSearchGlobal</code></td>
<td><code>search.global</code></td>
<td><code>N/search Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSearchRecord</code></td>
<td><code>search.load</code></td>
<td><code>search Module</code></td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td><code>nlapiSelectLineItem</code></td>
<td><code>Record.selectLine</code></td>
<td><code>N/record Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSelectNewLineItem</code></td>
<td><code>Record.selectNewLine</code></td>
<td><code>N/record Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSelectNode</code></td>
<td><code>XPath.select</code></td>
<td><code>N/xml Module</code></td>
<td>To mimic this functionality in SuiteScript 2.0, select a node with XPath.select(options) and then inspect the Node.textContent property.</td>
</tr>
<tr>
<td><code>nlapiSelectNodes</code></td>
<td><code>XPath.select</code></td>
<td><code>N/xml Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSelectValue</code></td>
<td><code>See Notes</code></td>
<td><code>N/xml Module</code></td>
<td>To mimic this functionality in SuiteScript 2.0, select an array of nodes with XPath.select(options) and then loop through each node's Node.textContent property.</td>
</tr>
<tr>
<td><code>nlapiSelectValues</code></td>
<td><code>See Notes</code></td>
<td><code>N/xml Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSendCampaignEmail</code></td>
<td><code>email.sendCampaignEvent</code></td>
<td><code>N/email Module</code></td>
<td>For a script sample, see <code>N/email Module Script Sample.</code></td>
</tr>
<tr>
<td><code>nlapiSendEmail</code></td>
<td><code>email.send</code></td>
<td><code>N/email Module</code></td>
<td>For a script sample, see <code>N/email Module Script Sample.</code></td>
</tr>
<tr>
<td><code>nlapiSendFax</code></td>
<td>N/A</td>
<td>N/A</td>
<td>This API does not have a SuiteScript 2.0 equivalent.</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Notes</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>-----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlapiSetCurrentLineItemDateValue(type, fieldId, dateTime)</td>
<td>Record.#setSublistValue(type, options)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
<td></td>
</tr>
<tr>
<td>nlapiSetCurrentLineItemDateValue(type, fieldId, lineNum, dateTime)</td>
<td>CurrentRecord.#setCurrentSublist#Value#(#options)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
<td></td>
</tr>
<tr>
<td>nlapiSetCurrentLineItemDateValue(type, fieldId, dateTime, timeZone)</td>
<td>CurrentRecord.#setCurrent#Record</td>
<td>Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
<td></td>
</tr>
<tr>
<td>nlapiSetCurrentLineItemText(fldnam, text)</td>
<td>CurrentRecord.#setText#(options)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
<td></td>
</tr>
<tr>
<td>nlapiSetCurrentLineItemTexts(fldnam, txts)</td>
<td>CurrentRecord.#setText#(options)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
<td></td>
</tr>
<tr>
<td>nlapiSetCurrentLineItemValues(fldnam, values)</td>
<td>CurrentRecord.#setValue#(options)</td>
<td>SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
<td></td>
</tr>
<tr>
<td>nlapiSetDateTimeValue(fieldId, dateTime, timeZone)</td>
<td>N/format Module</td>
<td>Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
<td></td>
</tr>
<tr>
<td>nlapiSetRedirectURL(type, identifier, id, editmode, parameters)</td>
<td>redirect.#redirect#(options)</td>
<td>For a script sample, see N/redirect Module Script Sample.</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
- SuiteScript 1.0 to SuiteScript 2.0 API Map – Functions (nlapi)
<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nlapiStringToDate(str, format)</code></td>
<td><code>format.#parse(options)</code></td>
<td>Format Module</td>
<td>For a script sample, see N/format Module Script Samples.</td>
</tr>
<tr>
<td><code>nlapiStringToXML(text)</code></td>
<td><code>Parser.fromString()</code></td>
<td>N/xml Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSubmitConfiguration(record, doSourcing, ignoreMandatoryFields)</code></td>
<td><code>Record.save(options)</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSubmitCSVImport(nlobjCSVImport)</code></td>
<td><code>N/task Module</code></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td><code>nlapiSubmitField(type, id, fields, values, doSourcing)</code></td>
<td><code>record.#submitFields()</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiSubmitFile(file)</code></td>
<td><code>File.save()</code></td>
<td>N/file Module</td>
<td>For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td><code>nlapiSubmitWorkflow(recordtype, id, workflowid, actionid, stateid)</code></td>
<td><code>workflow.#trigger(options)</code></td>
<td>N/workflow Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiValidateXML(xmlDocument, schemaDocument, schemaFolderId)</code></td>
<td><code>xml.#validate(options)</code></td>
<td>N/xml Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiViewCurrentLineItemSubrecord(sublist, fldname)</code></td>
<td><code>Record.#getCurrentSublistSubrecordSubrecord()</code></td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td><code>nlapiViewLineItemSubrecord(sublist, fldname, linenum)</code></td>
<td><code>Record.#getSublistSubrecord()</code></td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords. SuiteScript 2.0 begins sublist numbering with 0. SuiteScript 1.0 begins sublist numbering with 1.</td>
</tr>
<tr>
<td><code>nlapiViewSubrecord(fldname)</code></td>
<td><code>Record.#getSubrecord()</code></td>
<td>N/record Module</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td><code>nlapiVoidTransaction(transactionType, recordId)</code></td>
<td><code>transaction.#void(options)</code></td>
<td>N/transaction Module</td>
<td>For a script sample, see N/transaction Module Script Sample.</td>
</tr>
<tr>
<td><code>nlapiXMLToPDF(xmlstring)</code></td>
<td><code>render.#xmlToPdf(options)</code></td>
<td>N/render Module</td>
<td>Note that <code>TemplateRenderer.renderAsPdf()</code> is equivalent to <code>nlobjEmailMerger.#renderToString()</code>. For a script sample, see <code>N/render Module Script Sample</code>.</td>
</tr>
<tr>
<td><code>nlapi(XMLToString(xml)</code></td>
<td><code>Parser.#toString(options)</code></td>
<td>N/xml Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlapiYieldScript()</code></td>
<td>See Notes</td>
<td>See Notes</td>
<td>Note that the SuiteScript 2.0 Map/Reduce Script Type automatically incorporates yielding.</td>
</tr>
</tbody>
</table>

## SuiteScript 1.0 to SuiteScript 2.0 API Map – Objects (nlobj)

This topic maps SuiteScript 1.0 Objects (prefixed with “nlobj”) to their corresponding SuiteScript 2.0 APIs. Objects and their members are grouped alphabetically by object name. Each object has its own table containing all object members.
**Note:** NetSuite does not support calling SuiteScript 1.0 APIs from SuiteScript 2.0 scripts.

**Note:** To view a mapping of SuiteScript 1.0 Functions (prefixed with "nlapi") to their corresponding SuiteScript 2.0 APIs, see SuiteScript 1.0 to SuiteScript 2.0 API Map – Functions (nlapi).

- `nlobjAssistant`
- `nlobjButton`
- `nlobjButton`
- `nlobjColumn`
- `nlobjConfiguration`
- `nlobjContext`
- `nlobjCredentialBuilder`
- `nlobjCSVImport`
- `nlobjDuplicateJobRequest`
- `nlobjEmailMerger`
- `nlobjError`
- `nlobjField`
- `nlobjFieldGroup`
- `nlobjFile`
- `nlobjForm`
- `nlobjFuture`
- `nlobjJobManager`
- `nlobjList`
- `nlobjLogin`
- `nlobjMergeResult`
- `nlobjPortlet`
- `nlobjRecord`
- `nlobjRequest`
- `nlobjResponse`
- `nlobjSearch`
- `nlobjSearchColumn`
- `nlobjSearchFilter`
- `nlobjSearchResult`
- `nlobjSearchResultSet`
- `nlobjSelectOption`
- `nlobjSublist`
- `nlobjSubrecord`
- `nlobjTab`
- `nlobjTemplateRenderer`
<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjAssistant</td>
<td>serverWidget.Assistant</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.addField(name, type, label, source, group)</td>
<td>Assistant.addField(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.addFieldGroup(name, label)</td>
<td>Assistant.addFieldGroup(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.addStep(name, label)</td>
<td>Assistant.addStep(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.addSubList(name, type, label)</td>
<td>Assistant.addFieldGroup(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getAllFields()</td>
<td>Assistant.getFieldIds()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getAllFieldGroups()</td>
<td>Assistant.getFieldGroupIds()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getAllSteps()</td>
<td>Assistant.getSteps()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getAllSubLists()</td>
<td>Assistant.getSublistIds()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getCurrentStep()</td>
<td>Assistant.currentStep</td>
<td>N/ui/serverWidget Module</td>
<td>Note: currentStep is a property.</td>
</tr>
<tr>
<td>nlobjAssistant.getField(name)</td>
<td>Assistant.getField(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getFieldGroup(name)</td>
<td>Assistant.getFieldGroupOptions()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getLastAction()</td>
<td>Assistant.getLastAction()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getLastStep()</td>
<td>Assistant.getLastStep()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getNextStep()</td>
<td>Assistant.getNextStep()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getStep(name)</td>
<td>Assistant.getStep(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.getSubList(name)</td>
<td>Assistant.getSublistOptions()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.hasError()</td>
<td>Assistant.hasErrorHtml()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.isFinished()</td>
<td>Assistant.isFinished()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.sendRedirect(response)</td>
<td>Assistant.sendRedirect(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.setCurrentStep(step)</td>
<td>Assistant.currentStep</td>
<td>N/ui/serverWidget Module</td>
<td>Note: currentStep is a property.</td>
</tr>
<tr>
<td>nlobjAssistant.setError(html)</td>
<td>Assistant.errorHtml</td>
<td>N/ui/serverWidget Module</td>
<td>Note: errorHtml is a property.</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------</td>
<td>------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>nlobjAssistant.setFieldValues(...)</td>
<td>Assistant.setFieldValues(...)</td>
<td>Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.setFinished(html)</td>
<td>Assistant.finishedHtml</td>
<td>N/ui/serverWidget Module</td>
<td>Note that finishedHtml is a property.</td>
</tr>
<tr>
<td>nlobjAssistant.setNumbered(...)</td>
<td>Assistant.isNumbered()</td>
<td>N/ui/serverWidget Module</td>
<td>Note that hideStepNumber is a property.</td>
</tr>
<tr>
<td>nlobjAssistant.setOrdered(...)</td>
<td>Assistant.isNotOrdered</td>
<td>N/ui/serverWidget Module</td>
<td>Note that isNotOrdered is a property.</td>
</tr>
<tr>
<td>nlobjAssistant.setScript(script)</td>
<td>Assistant.setScript(script)</td>
<td>N/ui/serverWidget Module</td>
<td>Note that clientScriptFileId and clientScriptModulePath are properties. Use one of these SuiteScript 2.0 properties to attach an ad hoc client script to an assistant.</td>
</tr>
<tr>
<td>nlobjAssistant.setShortcut(show)</td>
<td>Assistant.setShortcut(show)</td>
<td>N/ui/serverWidget Module</td>
<td>Note that hideAddToShortcutsLink is a property.</td>
</tr>
<tr>
<td>nlobjAssistant.setSplash(title, text1, text2)</td>
<td>Assistant.setSplash(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistant.setTitle(title)</td>
<td>Assistant.title</td>
<td>N/ui/serverWidget Module</td>
<td>Note that title is a property.</td>
</tr>
</tbody>
</table>

**nlobjAssistantStep**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjAssistantStep</td>
<td>serverWidget.AssistantStep</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getAllFields()</td>
<td>Assistant.getFields()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getAllLineItemFields...</td>
<td>Assistant.getSublistFieldIds()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getAllLineItems()</td>
<td>Assistant.getSubmittedSublistIds()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getFieldValue(name)</td>
<td>Assistant.getValue()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getFieldValues(name)</td>
<td>Assistant.getValue()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getLineItemCount(...)</td>
<td>Assistant.getLineCount()</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getLineItemValue(group, name, line)</td>
<td>Auxiliary.getSublistValue...</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjAssistantStep.getStepNumber()</td>
<td>Assistant.stepNumber</td>
<td>N/ui/serverWidget Module</td>
<td>Note that stepNumber is a property.</td>
</tr>
<tr>
<td>nlobjAssistantStep.setHelpText(help)</td>
<td>Assistant.helpText</td>
<td>N/ui/serverWidget Module</td>
<td>Note that helpText is a property.</td>
</tr>
<tr>
<td>nlobjAssistantStep.setLabel(label)</td>
<td>Assistant.label</td>
<td>N/ui/serverWidget Module</td>
<td>Note that label is a property.</td>
</tr>
</tbody>
</table>
### nlobjButton

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjButton</td>
<td>serverWidget.Button</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjButton.disabled</td>
<td>isDisabled</td>
<td>N/ui/serverWidget Module</td>
<td>Note that isDisabled is a property.</td>
</tr>
<tr>
<td>nlobjButton.getLabel(label)</td>
<td>Button.label</td>
<td>N/ui/serverWidget Module</td>
<td>Note that label is a property.</td>
</tr>
<tr>
<td>nlobjButton.setVisible</td>
<td>isHidden</td>
<td>N/ui/serverWidget Module</td>
<td>Note that isHidden is a property.</td>
</tr>
</tbody>
</table>

### nlobjColumn

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjColumn</td>
<td>serverWidget.ListColumn</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjColumn.addParamToURL</td>
<td>ListColumn.addParamToURL</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>nlobjColumn.getLabel(label)</td>
<td>ListColumn.label</td>
<td>N/ui/serverWidget Module</td>
<td>Note that label is a property.</td>
</tr>
<tr>
<td>nlobjColumn.setURL(url, dynamic)</td>
<td>ListColumn.setURL(options)</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
</tbody>
</table>

### nlobjConfiguration

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjConfiguration</td>
<td>record.Record</td>
<td>N/record Module</td>
<td>Use the N/config Module method, config.load(options), to return a record.Record object. Then use the record.Record object members to access the specified configuration page. For a script sample, see N/config Module Script Sample.</td>
</tr>
<tr>
<td>nlobjConfiguration.getAllFields()</td>
<td>Record.getFields()</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjConfiguration.getField(fldnam)</td>
<td>Record.get(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjConfiguration.getFieldText(name)</td>
<td>Record.getText(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjConfiguration.getFieldTexts(name)</td>
<td>Record.getText(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjConfiguration.getFieldValue(name)</td>
<td>Record.getValue(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjConfiguration.getFieldValues(name)</td>
<td>Record.getValue(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjConfiguration.getType()</td>
<td>Record.type</td>
<td>N/record Module</td>
<td>Note that type is a property.</td>
</tr>
<tr>
<td>nlobjConfiguration.setText(text)</td>
<td>Record.setText(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td><code>nlobjConfiguration.setFieldTexts</code></td>
<td><code>Record.setText(options)</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjConfiguration.setFieldValue</code></td>
<td><code>Record.setValue(options)</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjConfiguration.setFieldValues</code></td>
<td><code>Record.setValue(options)</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
</tbody>
</table>

### nlobjContext

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nlobjContext</code></td>
<td><code>runtime.Script</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjContext</code></td>
<td><code>runtime.Session</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjContext</code></td>
<td><code>runtime.User</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjContext.getCompany()</code></td>
<td><code>runtime.accountId</code></td>
<td>N/runtime Module</td>
<td>Note that <code>accountId</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getDepartment()</code></td>
<td><code>User.department</code></td>
<td>N/runtime Module</td>
<td>Note that <code>department</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getDeploymentId()</code></td>
<td><code>Script.deploymentId</code></td>
<td>N/runtime Module</td>
<td>Note that <code>deploymentId</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getEmail()</code></td>
<td><code>User.email</code></td>
<td>N/runtime Module</td>
<td>Note that <code>email</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getEnvironment()</code></td>
<td><code>runtime.envType</code></td>
<td>N/runtime Module</td>
<td>Note that <code>envType</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getExecutionContext()</code></td>
<td><code>runtime.executionContext</code></td>
<td>N/runtime Module</td>
<td>Note that <code>executionContext</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getFeature(name)</code></td>
<td><code>runtime.isFeatureInEffect(options)</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjContext.getLocation()</code></td>
<td><code>User.location</code></td>
<td>N/runtime Module</td>
<td>Note that <code>location</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getLogLevel()</code></td>
<td><code>Script.logLevel</code></td>
<td>N/runtime Module</td>
<td>Note that <code>logLevel</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getName()</code></td>
<td><code>User.name</code></td>
<td>N/runtime Module</td>
<td>Note that <code>name</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getPercentComplete()</code></td>
<td><code>Script.percentComplete</code></td>
<td>N/runtime Module</td>
<td>Note that <code>percentComplete</code> is a property.</td>
</tr>
<tr>
<td>For a script sample, see N/runtime Module Script Sample.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>nlobjContext.getPermission(name)</code></td>
<td><code>User.getPermission(options)</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjContext.getPreference(name)</code></td>
<td><code>User.getPreference(options)</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjContext.getQueueCount()</code></td>
<td><code>runtime.queueCount</code></td>
<td>N/runtime Module</td>
<td>Note that <code>queueCount</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getRemainingUsage()</code></td>
<td><code>Script.getRemainingUsage()</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjContext.getRole()</code></td>
<td><code>User.role</code></td>
<td>N/runtime Module</td>
<td>Note that <code>role</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getRoleCenter()</code></td>
<td><code>User.roleCenter</code></td>
<td>N/runtime Module</td>
<td>Note that <code>roleCenter</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getRoleId()</code></td>
<td><code>User.roleId</code></td>
<td>N/runtime Module</td>
<td>Note that <code>roleId</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getScriptId()</code></td>
<td><code>Script.id</code></td>
<td>N/runtime Module</td>
<td>Note that <code>id</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjContext.getSessionObject(name)</code></td>
<td><code>Session.get(options)</code></td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
</tbody>
</table>
### SuiteScript 1.0 API to SuiteScript 2.0 API Map – Objects (nlobj)

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjContext.getSetting(type, name)</td>
<td>Script.getParameter(options)</td>
<td>runtime Module</td>
<td>The method <code>Script.getParameter(options)</code> is equivalent to <code>nlobjContext.getSetting('SCRIPT', name)</code>.</td>
</tr>
<tr>
<td></td>
<td>Session.get(options)</td>
<td></td>
<td>The method <code>Session.get(options)</code> is equivalent to <code>nlobjContext.getSetting('SESSION', name)</code>.</td>
</tr>
<tr>
<td></td>
<td>runtime.isFeatureInEffect(options)</td>
<td></td>
<td>The method <code>runtime.isFeatureInEffect(options)</code> is equivalent to <code>nlobjContext.getSetting('FEATURE', name)</code>.</td>
</tr>
<tr>
<td></td>
<td>User.getPermission(options)</td>
<td></td>
<td>The method <code>User.getPermission(options)</code> is equivalent to <code>nlobjContext.getSetting('PERMISSION', name)</code>.</td>
</tr>
<tr>
<td>nlobjContext.getSubsidiary()</td>
<td>User.subsidiary</td>
<td>N/runtime Module</td>
<td>Note that <code>subsidiary</code> is a property.</td>
</tr>
<tr>
<td>nlobjContext.getUser()</td>
<td>User.id</td>
<td>N/runtime Module</td>
<td>Note that <code>id</code> is a property.</td>
</tr>
<tr>
<td>nlobjContext.getVersion()</td>
<td>runtime.version</td>
<td>N/runtime Module</td>
<td>Note that <code>version</code> is a property.</td>
</tr>
<tr>
<td>nlobjContext.setPercentComplete(pct)</td>
<td>Script.percentComplete</td>
<td>N/runtime Module</td>
<td>Note that <code>percentComplete</code> is a property.</td>
</tr>
<tr>
<td>nlobjContext.setSessionObject(name, value)</td>
<td>Session.set(options)</td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjContext.setSetting(type, name, value)</td>
<td>Session.set(options)</td>
<td>N/runtime Module</td>
<td>-</td>
</tr>
</tbody>
</table>

### nlobjCredentialBuilder

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjCredentialBuilder(string, domainString)</td>
<td>SecureString</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.append(nlobjCredentialBuilder)</td>
<td>SecureString.appendSecureString(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.append(nlobjCredentialBuilder)</td>
<td>SecureString.appendString(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.base64()</td>
<td>SecureString.convertEncoding(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.md5()</td>
<td>SecureString.hash(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.replace(string1, string2)</td>
<td>SecureString.hash(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.sha1()</td>
<td>SecureString.hash(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.sha256()</td>
<td>SecureString.hash(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjCredentialBuilder.utf8()</td>
<td>SecureString.convertEncoding(options)</td>
<td>N/https Module</td>
<td>-</td>
</tr>
</tbody>
</table>
### nlobjCSVImport

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjCSVImport</td>
<td>task.CsvImportTask</td>
<td>N/task Module</td>
<td>Returned by task.create(options).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>var csvImpTaskObj = task.create({ taskType: task.TaskType.CSV_IMPORT, //Other Params });</td>
</tr>
<tr>
<td>nlobjCSVImport.setLinkedFile(sublist, file)</td>
<td>CsvImportTask.##linkedFiles</td>
<td>N/task Module</td>
<td>Note that linkedFiles is a property.</td>
</tr>
<tr>
<td>nlobjCSVImport.setMapping#(saved)</td>
<td>CsvImportTask.##mapping</td>
<td>N/task Module</td>
<td>Note that mappingId is a property.</td>
</tr>
<tr>
<td>nlobjCSVImport.setOption(option, value)</td>
<td>CsvImportTask.name</td>
<td>N/task Module</td>
<td>Note that name is a property.</td>
</tr>
<tr>
<td>nlobjCSVImport.setPrimaryFile(file)</td>
<td>CsvImportTask.##importFile</td>
<td>N/task Module</td>
<td>Note that importFile is a property.</td>
</tr>
<tr>
<td>nlobjCSVImport.setQueue(string)</td>
<td>CsvImportTask.queueId</td>
<td>N/task Module</td>
<td>Note that queueId is a property.</td>
</tr>
</tbody>
</table>

### nlobjDuplicateJobRequest

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobj##Duplicate##Job##Request</td>
<td>task.EntityDeduplicationTask</td>
<td>N/task Module</td>
<td>Returned by task.create(options).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>var dedupTaskObj = task.create({ taskType: task.TaskType.ENTITY_DEDUPLICATION, //Other Params });</td>
</tr>
<tr>
<td>nlobj##Duplicate##Job##Request.setEntityType#(entityType)</td>
<td>EntityDeduplicationTask.##entityType</td>
<td>N/task Module</td>
<td>Note that entityType is a property.</td>
</tr>
<tr>
<td>nlobj##Duplicate##Job##Request.setMasterId#(masterID)</td>
<td>EntityDeduplicationTask.##masterRecordId</td>
<td>N/task Module</td>
<td>Note that masterRecordId is a property.</td>
</tr>
<tr>
<td>nlobj##Duplicate##Job##Request.setMasterSelectionMode#(mode)</td>
<td>EntityDeduplicationTask.##masterSelectionMode</td>
<td>N/task Module</td>
<td>Note that masterSelectionMode is a property.</td>
</tr>
<tr>
<td>nlobj##Duplicate##Job##Request.setOperation#(operation)</td>
<td>EntityDeduplicationTask.##dedupeMode</td>
<td>N/task Module</td>
<td>Note that dedupeMode is a property.</td>
</tr>
<tr>
<td>nlobj##Duplicate##Job##Request.setRecords#(dupeRecords)</td>
<td>EntityDeduplicationTask.##recordIds</td>
<td>N/task Module</td>
<td>Note that recordIds is a property.</td>
</tr>
</tbody>
</table>
## nlobjEmailMerger

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjEmailMerger</td>
<td>render.EmailMerge</td>
<td>N/Render Module</td>
<td></td>
</tr>
<tr>
<td>nlobjEmailMerger.merge()</td>
<td></td>
<td>N/Render Module</td>
<td></td>
</tr>
<tr>
<td>nlobjEmailMerger.setCustomRecord(recordId)</td>
<td>SuiteNotes</td>
<td>N/Render Module</td>
<td>In SuiteScript 2.0, this value is set with a render.mergeEmail(options) parameter.</td>
</tr>
<tr>
<td>nlobjEmailMerger.setEntity(entityId)</td>
<td>SuiteNotes</td>
<td>N/Render Module</td>
<td>In SuiteScript 2.0, this value is set with a render.mergeEmail(options) parameter.</td>
</tr>
<tr>
<td>nlobjEmailMerger.setRecipient(recipientId)</td>
<td>SuiteNotes</td>
<td>N/Render Module</td>
<td>In SuiteScript 2.0, this value is set with a render.mergeEmail(options) parameter.</td>
</tr>
<tr>
<td>nlobjEmailMerger.setSupportCase(caseId)</td>
<td>SuiteNotes</td>
<td>N/Render Module</td>
<td>In SuiteScript 2.0, this value is set with a render.mergeEmail(options) parameter.</td>
</tr>
<tr>
<td>nlobjEmailMerger.setTransaction(transactionId)</td>
<td>SuiteNotes</td>
<td>N/Render Module</td>
<td>In SuiteScript 2.0, this value is set with a render.mergeEmail(options) parameter.</td>
</tr>
</tbody>
</table>

## nlobjError

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjError</td>
<td>error.SuiteScriptError</td>
<td>N/Error Module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>error.UserEventError</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nlobjError.getCode()</td>
<td></td>
<td>N/Error Module</td>
<td></td>
</tr>
<tr>
<td>nlobjError.getDetails()</td>
<td></td>
<td>N/Error Module</td>
<td></td>
</tr>
<tr>
<td>nlobjError.getId()</td>
<td></td>
<td>N/Error Module</td>
<td></td>
</tr>
<tr>
<td>nlobjError/InternalId()</td>
<td></td>
<td>N/Error Module</td>
<td></td>
</tr>
<tr>
<td>nlobjError.getStackTrace()</td>
<td></td>
<td>N/Error Module</td>
<td></td>
</tr>
<tr>
<td>nlobjError.getUserEvent()</td>
<td></td>
<td>N/Error Module</td>
<td></td>
</tr>
</tbody>
</table>

## nlobjField

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjField</td>
<td>serverWidget.Field</td>
<td>N/ui/serverWidget Module</td>
<td>Use the N/ui/serverWidget module to create and modify form fields in a Suitelet.</td>
</tr>
<tr>
<td></td>
<td>record.Field</td>
<td>N/Record Module</td>
<td>Use the N/Record module to access field metadata in client and server-side scripts.</td>
</tr>
</tbody>
</table>
### nlobjFieldGroup

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjFieldGroup</td>
<td>serverWidget.##FieldGroup</td>
<td>/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>N/ui/serverWidget Module</td>
<td></td>
<td></td>
</tr>
<tr>
<td>nlobjFieldGroup.setCollapsible(hidden)</td>
<td>FieldGroup.##isCollapsible</td>
<td>N/ui/serverWidget Module</td>
<td>Note that isCollapsible is a property.</td>
</tr>
<tr>
<td>nlobjFieldGroup.setLabel(label)</td>
<td>FieldGroup.label</td>
<td>N/ui/serverWidget Module</td>
<td>Note that label is a property.</td>
</tr>
<tr>
<td>nlobjFieldGroup.setShowBorder(show)</td>
<td>FieldGroup.##isBorderHidden</td>
<td>N/ui/serverWidget Module</td>
<td>Note that isBorderHidden is a property.</td>
</tr>
<tr>
<td>nlobjFieldGroup.setSingleColumn(column)</td>
<td>FieldGroup.##isSingleColumn</td>
<td>N/ui/serverWidget Module</td>
<td>Note that isSingleColumn is a property.</td>
</tr>
</tbody>
</table>

### nlobjFile

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjFile</td>
<td>file.File</td>
<td>N/file Module</td>
<td>For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td>nlobjFile.getDescription()</td>
<td>File.description</td>
<td>N/file Module</td>
<td>Note that description is a property.</td>
</tr>
<tr>
<td>nlobjFile.getFolder()</td>
<td>File.folder</td>
<td>N/file Module</td>
<td>Note that folder is a property. For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td>nlobjFile.getId()</td>
<td>File.id</td>
<td>N/file Module</td>
<td>Note that id is a property. For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td>nlobjFile.getName()</td>
<td>File.name</td>
<td>N/file Module</td>
<td>Note that name is a property. For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td>nlobjFile.getSize()</td>
<td>File.size</td>
<td>N/file Module</td>
<td>Note that size is a property.</td>
</tr>
<tr>
<td>nlobjFile.getType()</td>
<td>File.fileType</td>
<td>N/file Module</td>
<td>Note that fileType is a property. For a script sample, see N/file Module Script Sample. For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td>nlobjFile.getURL()</td>
<td>File.url</td>
<td>N/file Module</td>
<td>Note that url is a property.</td>
</tr>
<tr>
<td>nlobjFile.getValue()</td>
<td>File.getContents()</td>
<td>N/file Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjFile.isInactive()</td>
<td>File.isInactive</td>
<td>N/file Module</td>
<td>Note that isInactive is a property.</td>
</tr>
<tr>
<td>nlobjFile.isOnline()</td>
<td>File.isOnline</td>
<td>N/file Module</td>
<td>Note that isOnline is a property.</td>
</tr>
</tbody>
</table>
## nlobjFile

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nlobjFile.setDescription(description)</code></td>
<td><code>File.description</code></td>
<td>N/file Module</td>
<td>Note that <code>description</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjFile.setEncoding(encoding)</code></td>
<td><code>File.encoding</code></td>
<td>N/file Module</td>
<td>Note that <code>encoding</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjFile.setFolder(id)</code></td>
<td><code>File.folder</code></td>
<td>N/file Module</td>
<td>Note that <code>folder</code> is a property. You can also set the folder during file creation with <code>file.create(options)</code>. For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td><code>nlobjFile.setInactive(inactive)</code></td>
<td><code>File.isInactive</code></td>
<td>N/file Module</td>
<td>Note that <code>isInactive</code> is a property.</td>
</tr>
<tr>
<td><code>nlobjFile.setOnline(online)</code></td>
<td><code>File.isOnline</code></td>
<td>N/file Module</td>
<td>Note that <code>isOnline</code> is a property. For a script sample, see N/file Module Script Sample.</td>
</tr>
<tr>
<td><code>nlobjFile.setName(name)</code></td>
<td><code>File.name</code></td>
<td>N/file Module</td>
<td>Note that <code>name</code> is a property. For a script sample, see N/file Module Script Sample.</td>
</tr>
</tbody>
</table>

## nlobjForm

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nlobjForm</code></td>
<td><code>serverWidget.Form</code></td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjForm.addButton(name, label, script)</code></td>
<td><code>Form.addButton(options)</code></td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjForm.addCredentialField(id, label, website, scriptId, value, entityMatch, tab)</code></td>
<td><code>Form.addCredentialField(options)</code></td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjForm.addField(name, type, label, sourceOrRadio, tab)</code></td>
<td><code>Form.addField(options)</code></td>
<td>N/ui/serverWidget Module</td>
<td>For a script sample, see N/ui/serverWidget Module Script Samples</td>
</tr>
<tr>
<td><code>nlobjForm.addFieldGroup(name, label, tab)</code></td>
<td><code>Form.addFieldGroup(options)</code></td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjForm.addPageLink(type, title, url)</code></td>
<td><code>Form.addPageLink(options)</code></td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjForm.addResetButton(label)</code></td>
<td><code>Form.addResetButton(options)</code></td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlobjForm.addSubList(name, type, label, tab)</td>
<td>Form.addSublist(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>For a script sample, see N/â##ui/ serverWidget Module Script Samples</td>
</tr>
<tr>
<td>nlobjForm.addSubmitButton(label)</td>
<td>Form.addSubmitButton(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>For a script sample, see N/â##ui/ serverWidget Module Script Samples</td>
</tr>
<tr>
<td>nlobjForm.addSubTab(name, label, tab)</td>
<td>Form.addSubtab(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.addTab(name, label)</td>
<td>Form.addTab(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.getField(name, radio)</td>
<td>Form.getField(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.getSubList(name)</td>
<td>Form.getSublist(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.getSubTab(name)</td>
<td>Form.getSubtab(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.getTab(name)</td>
<td>Form.getTab(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.getTabs()</td>
<td>Form.getTabs()</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.insertField(field, nextfld)</td>
<td>Form.insertField(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.insertSubList(sublist, nextsub)</td>
<td>Form.â##insertSublist##(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.insertSubTab(subtab, nextsub)</td>
<td>Form.â##insertSubtab##(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.insertTab(tab, nexttab)</td>
<td>Form.insertTab(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.removeButton(name)</td>
<td>Form.â##removeButton##(options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjForm.setFieldValues(values)</td>
<td>Form.â##updateDefault##(##options)</td>
<td>N/â##ui/ â##serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>----------------------</td>
<td>-----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlobjForm.setScript(script)</td>
<td>Form.###clientScriptFileId Form.###clientScriptModulePath</td>
<td>N/ai/ ###serverWidget Module</td>
<td>Note that clientScriptFileId and clientScriptModulePath are properties. Use one of these SuiteScript 2.0 properties to attach an ad hoc client script to a form.</td>
</tr>
<tr>
<td>nlobjForm.setTitle(title)</td>
<td>Form.title</td>
<td>N/ai/ ###serverWidget Module</td>
<td>Note that title is a property.</td>
</tr>
</tbody>
</table>

**nlobjFuture**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjFuture</td>
<td>task.###Entity###Deduplication###Task###Status</td>
<td>N/task Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjFuture.isCancelled()</td>
<td>Entity###Deduplication###Task###Status ###status</td>
<td>N/task Module</td>
<td>Note that status is a property.</td>
</tr>
<tr>
<td>nlobjFuture.isDone()</td>
<td>Entity###Deduplication###Task###Status ###status</td>
<td>N/task Module</td>
<td>Note that status is a property.</td>
</tr>
</tbody>
</table>

**nlobjJobManager**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjJobManager</td>
<td>-</td>
<td>N/task Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjJobManager createJobRequest()</td>
<td>-</td>
<td>N/task Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjJobManager.getFuture()</td>
<td>-</td>
<td>N/task Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjJobManager.submit###(nlobjDuplicateJobRequest)</td>
<td>-</td>
<td>N/task Module</td>
<td>-</td>
</tr>
</tbody>
</table>

**nlobjList**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjList</td>
<td>serverWidget.List</td>
<td>N/ai/ ###serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjList.addButton(name, label, script)</td>
<td>List.addButton(options)</td>
<td>N/ai/ ###serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjList.addColumn(name, type, label, align)</td>
<td>List.addColumn(options)</td>
<td>N/ai/ ###serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjList.addEditColumn###(column, showView, showHrefCol)</td>
<td>List.###addEditColumn###(options)</td>
<td>N/ai/ ###serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlobjList.addPageLink(type, title, url)</td>
<td>List.addPageLink(options)</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjList.addRow(row)</td>
<td>List.addRow(options)</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjList.addRow(rows)</td>
<td>List.addRow(options)</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjList.setScript(script)</td>
<td>List.clientScriptFileId</td>
<td>N/ui/serverWidget Module</td>
<td>Note that clientScriptFileId and clientScriptModulePath are properties. Use one of these SuiteScript 2.0 properties to attach an ad hoc client script to a form.</td>
</tr>
<tr>
<td>nlobjList.setStyle(style)</td>
<td>List.style</td>
<td>N/ui/serverWidget Module</td>
<td>Note that style is a property.</td>
</tr>
<tr>
<td>nlobjList.setTitle(title)</td>
<td>List.title</td>
<td>N/ui/serverWidget Module</td>
<td>Note that title is a property.</td>
</tr>
</tbody>
</table>

**nlobjLogin**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjLogin</td>
<td>See Notes.</td>
<td>N/auth Module</td>
<td>See nlobjLogin members.</td>
</tr>
<tr>
<td>nlobjLogin.changeEmail(currentPassword, newEmail, justThisAccount)</td>
<td>auth.changeEmail(options)</td>
<td>N/auth Module</td>
<td>For a script sample, see N/auth Module Script Sample.</td>
</tr>
<tr>
<td>nlobjLogin.changePassword(currentPassword, newPassword)</td>
<td>auth.changePassword(options)</td>
<td>N/auth Module</td>
<td>For a script sample, see N/auth Module Script Sample.</td>
</tr>
</tbody>
</table>

**nlobjMergeResult**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjMergeResult</td>
<td>render.EmailMergeResult</td>
<td>N/render Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjMergeResult.getBody()</td>
<td>EmailMergeResult.body</td>
<td>N/render Module</td>
<td>Note that body is a property.</td>
</tr>
<tr>
<td>nlobjMergeResult.getSubject()</td>
<td>EmailMergeResult.subject</td>
<td>N/render Module</td>
<td>Note that subject is a property.</td>
</tr>
</tbody>
</table>
## nlobjPortlet

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjPortlet</td>
<td>Portlet Object</td>
<td>See Notes</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.addColumn(name, type, label, just)</td>
<td>Portlet.addColumn(options)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.addEditColumn(showView, showHrefCol)</td>
<td>Portlet.addEditColumn(options)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.addField(name, type, label, source)</td>
<td>Portlet.addField(options)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.addLine(text, url, indent)</td>
<td>Portlet.addLine(options)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.addRow(row)</td>
<td>Portlet.addRow(options)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.addRows(rows)</td>
<td>Portlet.addRows(options)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.setHtml(html)</td>
<td>Portlet.html</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.setRefreshInterval(n)</td>
<td>Portlet.setRefreshInterval(n)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.setScript(scriptid)</td>
<td>Portlet.setScript(scriptid)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.setSubmitButton(url, label, target)</td>
<td>Portlet.setSubmitButton(options)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>nlobjPortlet.setTitle(title)</td>
<td>Portlet.setTitle(title)</td>
<td>-</td>
<td>For additional information, see the help topic SuiteScript 2.0 Portlet Script Type.</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-------------------------------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>nlobjRecord</td>
<td>record.Record</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>currentRecord.CurrentRecord</td>
<td>N/currentRecord Module</td>
<td></td>
</tr>
<tr>
<td>nlobjRecord.commitLineItem(group, ignoreRecalc)</td>
<td>Record.commitLine(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CurrentRecord.commitLine(options)</td>
<td>N/currentRecord Module</td>
<td></td>
</tr>
<tr>
<td>nlobjRecord.createSubrecord(fldname)</td>
<td>record.getSubrecord(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CurrentRecord.getSubrecord(options)</td>
<td>N/currentRecord Module</td>
<td></td>
</tr>
<tr>
<td>nlobjRecord.editSubrecord(fldname)</td>
<td>record.getSubrecord(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CurrentRecord.getSubrecord(options)</td>
<td>N/currentRecord Module</td>
<td></td>
</tr>
<tr>
<td>nlobjRecord.findLineItemValue(group, fldnam, column, val)</td>
<td>record.findSublistLineWithValue(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td></td>
<td>CurrentRecord.findSublistLineWithValue(options)</td>
<td>N/currentRecord Module</td>
<td></td>
</tr>
<tr>
<td>nlobjRecord.getAllLineItemFields(group)</td>
<td>record.getSublistFields(options)</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjRecord.getAllFields()</td>
<td>Record.getFields()</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>nlobjRecord.getDateAndTime(fieldId, timeZone)</td>
<td>CurrentRecord.getDateAndTimeValue(type, fieldId, timeZone)</td>
<td>N/format Module</td>
<td>Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
</tbody>
</table>

Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.
<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nlobjRecord.getLineItemText(group, fldnam, linenum)</code></td>
<td><code>CurrentRecord.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemValue(group, name, linenum)</code></td>
<td><code>CurrentRecord.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemMatrixField(group, fldnam, linenum, column)</code></td>
<td><code>CurrentRecord.##getMatrixSublistField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemMatrixValue(group, fldnam, linenum, column)</code></td>
<td><code>CurrentRecord.##getMatrixSublistValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemText(group, fldnam, linenum)</code></td>
<td><code>CurrentRecord.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemValue(group, name, linenum)</code></td>
<td><code>CurrentRecord.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemCount(group)</code></td>
<td><code>Record.##getLineCount(options)</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemField(group, fldnam, linenum)</code></td>
<td><code>Record.##getSublistField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemMatrixField(group, fldnam, linenum, column)</code></td>
<td><code>CurrentRecord.##getMatrixSublistField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemMatrixValue(group, fldnam, linenum, column)</code></td>
<td><code>CurrentRecord.##getMatrixSublistValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getField(name)</code></td>
<td><code>Record.##getField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldText(name)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldTexts(name)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValue(name)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValues(name)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getDateTimeValue(fieldId, timeZone)</code></td>
<td><code>See Notes</code></td>
<td>N/format Module</td>
<td>Method returns an array for multi-select fields. Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
<tr>
<td><code>nlobjRecord.getField(fieldId)</code></td>
<td><code>Record.##getField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldText(fieldId)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldTexts(fieldId)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValue(fieldId)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValues(fieldId)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getId()</code></td>
<td><code>Record.id</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemCount(group)</code></td>
<td><code>Record.##getLineCount(options)</code></td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemField(group, fldnam, linenum)</code></td>
<td><code>Record.##getSublistField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemMatrixField(group, fldnam, linenum, column)</code></td>
<td><code>CurrentRecord.##getMatrixSublistField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemMatrixValue(group, fldnam, linenum, column)</code></td>
<td><code>CurrentRecord.##getMatrixSublistValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemText(group, fldnam, linenum)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemValue(group, name, linenum)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getField(fldnam)</code></td>
<td><code>Record.##getField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldText(fldnam)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldTexts(fldnam)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValue(fldnam)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValues(fldnam)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getDateTimeValue(fieldId, timeZone)</code></td>
<td><code>See Notes</code></td>
<td>N/format Module</td>
<td>Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
<tr>
<td><code>nlobjRecord.getField(fieldId)</code></td>
<td><code>Record.##getField(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldText(fieldId)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldTexts(fieldId)</code></td>
<td><code>Record.##getText(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValue(fieldId)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.getFieldValues(fieldId)</code></td>
<td><code>Record.##getValue(options)</code></td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>------------------------</td>
<td>-------</td>
</tr>
<tr>
<td><code>nlobjRecord.getLineItemValues(type, fldnam, linenum)</code></td>
<td><code>Record.getLineItemValue(options)</code></td>
<td><code>CurrentRecord.getLineItemValue</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.getMatrixCount(group, fldnam)</code></td>
<td><code>Record.getMatrixHeaderCount(options)</code></td>
<td><code>CurrentRecord.getMatrixHeaderCount</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.getMatrixField(group, fldname, column)</code></td>
<td><code>Record.getMatrixHeaderField(options)</code></td>
<td><code>CurrentRecord.getMatrixHeaderField</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.getMatrixValue(group, fldnam, column)</code></td>
<td><code>Record.getMatrixHeaderValue(options)</code></td>
<td><code>CurrentRecord.getMatrixHeaderValue</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.getRecordType()</code></td>
<td><code>Record.type</code></td>
<td><code>N/record Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.insertLineItem(group, linenum, ignoreRecalc)</code></td>
<td><code>Record.insertLine(options)</code></td>
<td><code>CurrentRecord.insertLine</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.removeLineItem(group, linenum, ignoreRecalc)</code></td>
<td><code>Record.removeLine(options)</code></td>
<td><code>CurrentRecord.removeLine</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.removeCurrentLineItemSubrecord(sublist, fldname)</code></td>
<td><code>Record.removeCurrentSublistSubrecord</code></td>
<td><code>N/record Module</code></td>
<td>-</td>
</tr>
<tr>
<td><code>nlobjRecord.removeSubrecord(fldname)</code></td>
<td><code>Record.removeSubrecord(options)</code></td>
<td><code>CurrentRecord.removeSubrecord</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.selectLineItem(group, linenum)</code></td>
<td><code>Record.selectLine(options)</code></td>
<td><code>CurrentRecord.selectLine</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.selectNewLineItem(group)</code></td>
<td><code>Record.selectNewLine(options)</code></td>
<td><code>CurrentRecord.selectNewLine</code></td>
<td><code>N/record Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.setCurrentLineItemDateValue(type, fieldId, dateTime, timeZone)</code></td>
<td><code>See Notes</code></td>
<td><code>N/format Module</code></td>
<td>Use the <code>N/format</code> module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
<tr>
<td><code>nlobjRecord.setCurrentLineItemMatrixValue(group, fldnam, column, value)</code></td>
<td><code>Record.setCurrentMatrixSublistValue</code></td>
<td><code>CurrentRecord.setCurrentMatrixSublistValue</code></td>
<td><code>N/currentRecord Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.setCurrentLineItemValue(group, name, value)</code></td>
<td><code>Record.setCurrentSublistValue</code></td>
<td><code>CurrentRecord.setCurrentSublistValue</code></td>
<td><code>N/currentRecord Module</code></td>
</tr>
<tr>
<td><code>nlobjRecord.setDateTimeValue(fieldId, dateTime, timeZone)</code></td>
<td><code>See Notes</code></td>
<td><code>N/format Module</code></td>
<td>Use the <code>N/format</code> module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
</tbody>
</table>
### SuiteScript 1.0 API vs. SuiteScript 2.0 API

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjRecord.setFieldText(name, text)</td>
<td>Record.##setText(options)</td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRecord.setFieldTexts(name, text)</td>
<td>Record.##setText(options)</td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRecord.setFieldValue(name, value)</td>
<td>Record.##setValue(options)</td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRecord.setFieldValues(name, value)</td>
<td>Record.##setValue(options)</td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRecord.setLineItemDateValue(type, fieldId, lineNum, dateTime, timeZone)</td>
<td>See Notes</td>
<td>N/format Module</td>
<td>Use the N/format module to mimic this functionality in SuiteScript 2.0.</td>
</tr>
<tr>
<td>nlobjRecord.setLineItemValue(group, name, linenum, value)</td>
<td>Record.##setSublistValue(group, name, linenum, value)</td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRecord.setMatrixValue(group, fldnam, column, value)</td>
<td>Record.##setMatrixHeaderValue(group, fldnam, column, value)</td>
<td>N/currentRecordModule</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRecord.viewCurrentLineItemSubrecord(sublist, fldname)</td>
<td>Record.##getCurrentSublistSubrecord(sublist, fldname)</td>
<td>N/currentRecordModule</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlobjRecord.viewLineItemSubrecord(sublist, fldname, linenum)</td>
<td>Record.##getSublistSubrecord(sublist, fldname, linenum)</td>
<td>N/currentRecordModule</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
<tr>
<td>nlobjRecord.viewSubrecord(fldname)</td>
<td>Record.##getSubrecord(fldname)</td>
<td>N/currentRecordModule</td>
<td>Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.</td>
</tr>
</tbody>
</table>

### nlobjRequest

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjRequest</td>
<td>http.ServerRequest</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRequest.getAllHeaders()</td>
<td>ServerResponse.##writePage(options)</td>
<td>N/http Module N/https Module</td>
<td>ServerResponse.##headers is read-only.</td>
</tr>
</tbody>
</table>
## SuiteScript 1.0 API to SuiteScript 2.0 API Map – Objects (nlobj)

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjRequest.getAllParameters()</td>
<td>ServerRequest.parameters</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.##parameters(options) is read-only.</td>
</tr>
<tr>
<td>nlobjRequest.getBody()</td>
<td>ServerRequest.parameters</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.##body(options) is read-only.</td>
</tr>
<tr>
<td>nlobjRequest.getFile(id)</td>
<td>ServerRequest.files</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.files(options) is read-only.</td>
</tr>
<tr>
<td>nlobjRequest.getHeader(name)</td>
<td>ServerResponse.##writePage(options)</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.##headers(options) is read-only.</td>
</tr>
<tr>
<td>nlobjRequest.getLineItemCount(group)</td>
<td>ServerResponse.##writePage(options)</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRequest.getLineItemValue(group, name, line)</td>
<td>ServerResponse.##getSublistValue(options)</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjRequest.getMethod()</td>
<td>ServerRequest.method</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.##method(options) is read-only.</td>
</tr>
<tr>
<td>nlobjRequest.getParameter(name)</td>
<td>ServerRequest.parameters</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.##parameters(options) is read-only.</td>
</tr>
<tr>
<td>nlobjRequest.getParameterValues(name)</td>
<td>ServerRequest.parameters</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.##parameters(options) is read-only.</td>
</tr>
<tr>
<td>nlobjRequest.getURL()</td>
<td>ServerRequest.method</td>
<td>N/http Module N/https Module</td>
<td>ServerRequest.url is read-only.</td>
</tr>
</tbody>
</table>

### nlobjResponse

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjResponse</td>
<td>http.ServerResponse</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.addHeader(name, value)</td>
<td>ServerResponse.##addHeader(options)</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.getAllHeaders()</td>
<td>ServerResponse.##getHeader(options)</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.getHeaders(name)</td>
<td>ServerResponse.##getHeader(options)</td>
<td>N/http Module N/https Module</td>
<td>If multiple values are assigned to the header name, serverResponse.##getHeader(options) returns the values as an Array.</td>
</tr>
<tr>
<td>nlobjResponse.getBody()</td>
<td>-</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.getCode()</td>
<td>-</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.getError()</td>
<td>-</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.getHeader(name)</td>
<td>ServerResponse.##getHeader(options)</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.getHeaders(name)</td>
<td>ServerResponse.##getHeader(options)</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.renderPDF(xmlString)</td>
<td>ServerResponse.##renderPdf(options)</td>
<td>N/http Module N/https Module</td>
<td>-</td>
</tr>
</tbody>
</table>
### nlobjResponse

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjResponse.setCDNCacheable(type)</td>
<td>response.setCDNCacheable(type)</td>
<td>N/http Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.setContentType(type, name, disposition)</td>
<td>N/http Module</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>nlobjResponse.setEncoding(encodingType)</td>
<td>N/http Module</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>nlobjResponse.setHeader(name, value)</td>
<td>ServerResponse.setHeader(options)</td>
<td>N/http Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.sendRedirect(type, identifier, id, editmode, parameters)</td>
<td>ServerResponse.setHeader(options)</td>
<td>N/http Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.write(output)</td>
<td>ServerResponse.write(options)</td>
<td>N/http Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.writeLine(output)</td>
<td>ServerResponse.write(options)</td>
<td>N/http Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjResponse.writePage(pageObject)</td>
<td>ServerResponse.writePage(options)</td>
<td>N/http Module</td>
<td>-</td>
</tr>
</tbody>
</table>

### nlobjSearch

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSearch</td>
<td>search.Search</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.addColumn(column)</td>
<td>Search.columns</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.addColumns(columns)</td>
<td>Search.columns</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.addFilter(filter)</td>
<td>Search.columns</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.addFilters(filters)</td>
<td>Search.columns</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.deleteSearch()</td>
<td>search.delete(options)</td>
<td>N/search Module</td>
<td>For a script sample, see N/search Module Script Samples.</td>
</tr>
<tr>
<td>nlobjSearch.getColumns()</td>
<td>Search.columns</td>
<td>N/search Module</td>
<td>Note that columns is a property.</td>
</tr>
<tr>
<td>nlobjSearch.getFilterExpression()</td>
<td>Search.filterExpression</td>
<td>N/search Module</td>
<td>Note that filterExpression is a property.</td>
</tr>
<tr>
<td>nlobjSearch.getFilters()</td>
<td>Search.filters</td>
<td>N/search Module</td>
<td>Note that filters is a property.</td>
</tr>
<tr>
<td>nlobjSearch.getId()</td>
<td>Search.searchId</td>
<td>N/search Module</td>
<td>Note that id is a property.</td>
</tr>
<tr>
<td>nlobjSearch.getIsPublic()</td>
<td>Search.isPublic</td>
<td>N/search Module</td>
<td>Note that isPublic is a property.</td>
</tr>
<tr>
<td>nlobjSearch.getScriptId()</td>
<td>Search.id</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.getSearchType()</td>
<td>Search.searchType</td>
<td>N/search Module</td>
<td>Note that searchType is a property.</td>
</tr>
<tr>
<td>nlobjSearch.runSearch()</td>
<td>Search.run()</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.saveSearch(title, scriptId)</td>
<td>Search.save()</td>
<td>N/search Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSearch.setColumns(columns)</td>
<td>Search.columns</td>
<td>N/search Module</td>
<td>Note that columns is a property.</td>
</tr>
<tr>
<td>SuiteScript 1.0 API</td>
<td>SuiteScript 2.0 API</td>
<td>SuiteScript 2.0 Module</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>--------------------------------------</td>
<td>-----------------------------</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>nlobjSearch.setFilterExpression(filterExpression)</td>
<td>n/search Module</td>
<td>Note that filterExpression is a property.</td>
<td></td>
</tr>
<tr>
<td>nlobjSearch.setFilters(filters)</td>
<td>Search.filters</td>
<td>N/search Module</td>
<td>Note that filters is a property.</td>
</tr>
<tr>
<td>nlobjSearch.setIsPublic(type)</td>
<td>Search.isPublic</td>
<td>N/Search Module</td>
<td>Note that isPublic is a property.</td>
</tr>
<tr>
<td>nlobjSearch.setRedirectURLToSearch()</td>
<td>redirect.toSearch(options)</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>nlobjSearch.setRedirectURLToSavedSearch()</td>
<td>redirect.toSavedSearchResult(options)</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

**nlobjSearchColumn**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSearchColumn</td>
<td>search.Column</td>
<td>N/search Module</td>
<td>-</td>
</tr>
</tbody>
</table>

**nlobjSearchFilter**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSearchFilter</td>
<td>search.Filter</td>
<td>N/search Module</td>
<td>-</td>
</tr>
</tbody>
</table>

**nlobjSearchResult**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSearchResult</td>
<td>search.Result</td>
<td>N/search Module</td>
<td>-</td>
</tr>
</tbody>
</table>

**nlobjSearchResultSet**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSearchResultSet</td>
<td>search.ResultSet</td>
<td>N/search Module</td>
<td>-</td>
</tr>
</tbody>
</table>

**nlobjSelectOption**

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSelectOption</td>
<td>See Notes</td>
<td>N/record Module</td>
<td>See mapping for nlobjSelectOption methods.</td>
</tr>
<tr>
<td>nlobjSelectOption.getId()</td>
<td></td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>N/currentRecord Module</td>
<td></td>
</tr>
</tbody>
</table>
### nlobjSelectOption

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSelectOption.getText()</td>
<td>N/record:## Field.##getSelectOptions##(options)</td>
<td>N/record Module</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>N/currentRecord:## Field.##getSelectOptions##(options)</td>
<td>N/currentRecord Module</td>
<td>-</td>
</tr>
</tbody>
</table>

### nlobjSublist

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSubList</td>
<td>serverWidget.Sublist</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.addButton(name, label, script)</td>
<td>Sublist.addButton(options)</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.addField(name, type, label, source)</td>
<td>Sublist.addField(options)</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.addMarkAllButtons()</td>
<td>Sublist.addMarkAllButtons()</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.addRefreshButton()</td>
<td>Sublist.addRefreshButton()</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.getLineItemCount()</td>
<td>Sublist.lineCount</td>
<td>N/ui/serverWidget Module</td>
<td>Note that lineCount is a property</td>
</tr>
<tr>
<td>nlobjSubList.getLineItemValue(group, fldnam, linenum)</td>
<td>Sublist.##getSublistValue##(options)</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.setAmountField(field)</td>
<td>-</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubListsetDisplayType(type)</td>
<td>Sublist.displayType</td>
<td>N/ui/serverWidget Module</td>
<td>Note that displayType is a property</td>
</tr>
<tr>
<td>nlobjSubList.setHelpText(help)</td>
<td>Sublist.helpText</td>
<td>N/ui/serverWidget Module</td>
<td>Note that helpText is a property</td>
</tr>
<tr>
<td>nlobjSubList.setLabel(label)</td>
<td>Sublist.label</td>
<td>N/ui/serverWidget Module</td>
<td>Note that label is a property</td>
</tr>
<tr>
<td>nlobjSubList.setLineItemValue(name, linenum, value)</td>
<td>Sublist.##setSublistValue##(options)</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.setLineItemValues(values)</td>
<td>-</td>
<td>N/ui/serverWidget Module</td>
<td>-</td>
</tr>
<tr>
<td>nlobjSubList.setUniqueField(name)</td>
<td>Sublist.##updateUniqueFieldId##(options)</td>
<td>N/ui/serverWidget Module</td>
<td>Note that uniqueFieldId is a property</td>
</tr>
</tbody>
</table>

### nlobjSubrecord

<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjSubrecord</td>
<td>See Notes</td>
<td>N/record Module</td>
<td>SuiteScript 2.0 subrecords are returned as record.Record objects.</td>
</tr>
</tbody>
</table>
### SuiteScript 1.0 API | SuiteScript 2.0 API | SuiteScript 2.0 Module | Notes
--- | --- | --- | ---
**SuiteScript 1.0 API** | **SuiteScript 2.0 API** | **SuiteScript 2.0 Module** | **Notes**
**nlobjSubrecord**
| cancel() | See Notes | N/record Module | SuiteScript 2.0 subrecords are returned as record.Record objects. Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.
| commit() | See Notes | N/record Module | This API does not have a SuiteScript 2.0 equivalent. SuiteScript 2.0 subrecords are returned as record.Record objects. Note that scripting subrecords in SuiteScript 2.0 is fundamentally different from scripting subrecords in SuiteScript 1.0. For additional information, see the SuiteScript 2.0 topics under SuiteScript 2.0 Scripting Subrecords.

### nlobjTab

| SuiteScript 1.0 API | SuiteScript 2.0 API | SuiteScript 2.0 Module | Notes |
--- | --- | --- | ---
nlobjTab | serverWidget.Tab | N/ui/serverWidget Module | - |
nlobjTab.setLabel(label) | Tab.label | N/ui/serverWidget Module | Note that label is a property |
nlobjTab.setHelpText(help) | Tab.helpText | N/ui/serverWidget Module | Note that helpText is a property |

### nlobjTemplateRenderer

| SuiteScript 1.0 API | SuiteScript 2.0 API | SuiteScript 2.0 Module | Notes |
--- | --- | --- | ---
nlobjTemplateRenderer | render.TemplateRenderer | N/render Module | For a script sample, see N/render Module Script Sample. |
nlobjTemplateRenderer.addRecord(var, record) | TemplateRenderer.addRecord(options) | N/render Module | For a script sample, see N/render Module Script Sample. |
nlobjTemplateRenderer.addSearchResults(var, searchResult) | TemplateRenderer.addSearchResults(options) | N/render Module | For a script sample, see N/render Module Script Sample. |
nlobjTemplateRenderer.renderToResponse() | TemplateRenderer.renderToResponse() | N/render Module | - |
nlobjTemplateRenderer.renderToString() | TemplateRenderer.renderAsString() | N/render Module | - |
<table>
<thead>
<tr>
<th>SuiteScript 1.0 API</th>
<th>SuiteScript 2.0 API</th>
<th>SuiteScript 2.0 Module</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlobjTemplateRenderer.setTemplate(template)</td>
<td>TemplateRenderer.#templateContent</td>
<td>N/render Module</td>
<td>For a script sample, see N/render Module Script Sample.</td>
</tr>
</tbody>
</table>
SuiteScript 2.0 Global Objects

SuiteScript 2.0 includes the following global objects. You can use these objects in your scripts without loading them as dependencies.

- define Object
- require Function
- log Object
- util Object
- toString()
- JSON object
- Promise Object

Note: In JavaScript, all functions are objects. The define Object and require Function topics discuss the define() and require() functions used by SuiteScript 2.0 to load and define modules.

define Object

The define object is an overloaded function that is used to create entry point scripts and custom modules in SuiteScript 2.0. This function executes asynchronously on the client side and synchronously on the server side. The define object conforms to the Asynchronous Module Definition (AMD) specification.

Note: An overloaded function has multiple signatures. A signature is the function name and all available parameters.

SuiteScript 2.0 supports the following define() signatures:

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>define(moduleObject)</td>
<td>object</td>
<td>Returns a module object based on the supplied moduleObject argument. The moduleObject argument can be any JavaScript object, including a function. Use this define() signature if your entry point script or custom module requires no dependencies.</td>
</tr>
<tr>
<td>Function</td>
<td>define(id, [dependencies,] moduleObject)</td>
<td>object</td>
<td>Loads all dependencies and then executes the supplied callback function. Returns a module object based on the callback.</td>
</tr>
</tbody>
</table>

Use the define() function to do the following:

- Create a SuiteScript script file. Load the required dependent modules and define the functionality for the SuiteScript script type in the callback function. The return statement in the callback function must include at least one entry point and entry point function. All entry points must belong to the same script type.
Any implementation of a SuiteScript script type that returns an entry point must use the define() Function.

- Create and return a custom module. You can then include the custom module as dependency in another script. Use the `define(id, [dependencies,] moduleObject)` signature if your module requires dependencies. If the custom module does not require any dependencies, use the `define(moduleObject)` signature.

For more information about custom modules, see the help topic SuiteScript 2.0 Custom Modules.
For more information about entry points, see the help topic SuiteScript 2.0 Script Types.

**define() Function Guidelines**

Use the following guidelines with the define() Function:

- SuiteScript API calls can be executed only after the define callback's return statement has executed. Consequently, you cannot use native SuiteScript 2.0 module methods when you create a custom module. You can make SuiteScript API calls after the Module Loader creates and loads the custom module.
- If you need to debug your code on demand in the NetSuite Debugger, you must use a require() Function. The NetSuite Debugger cannot step though a define() Function.
- Any dependencies used in the define() Function are loaded before the callback function executes.
- You can load only modules that are stored in the NetSuite file cabinet. Do not attempt to import scripts via HTTP/S.

For example, if given `define(['http://somewebsite.com/purchaseTotal.js'], function(purchaseTotal){...});`, the `purchaseTotal` dependency is not valid.

**define(moduleObject)**

**Description**

Function used to create entry point scripts and custom modules in SuiteScript 2.0. For more information, see the help topics SuiteScript 2.0 Entry Point Script Creation and Deployment and SuiteScript 2.0 Custom Modules.

Use this `define()` signature if your entry point script or custom module requires no dependencies.

If you are creating an entry point script, the define() function must return an object consisting of at least one key/value pair. Each key must be an entry point and the corresponding value must be a named entry point function. All entry points must be for the same script type. Your script can have only one entry point script and the entry point script must be only one script type.

**Returns**

Object

**Global object**

`define Object`

**Since**

Version 2015 Release 2

**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>moduleObject</td>
<td>Object</td>
<td>Required</td>
<td>A callback function or a module object</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>
define Object

Syntax

The following code snippets show sample syntax for the define(moduleObject) function signature. These snippets are not functional examples or a complete list.

Define a Function

```
// lib.js
define({
    test: function () {
        return true;
    }
});
```

OR

```
// lib.js
define(function () {
    return true
});
```

Define an object

```
// lib.js
define({
    color: "black",
    size: "unisize"
});
```

Define a Primitive Value

```
// lib.js
define("test");
```

define(id, [dependencies,] moduleObject)

<table>
<thead>
<tr>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function used to create entry point scripts and custom modules in SuiteScript 2.0. For more information, see the help topics SuiteScript 2.0 Entry Point Script Creation and Deployment and SuiteScript 2.0 Custom Modules.</td>
</tr>
<tr>
<td>If you are creating an entry point script, the define() function must return an object consisting of at least one key/value pair. Each key must be an entry point and the corresponding value must be a named entry point function. All entry points must be for the same script type. Your script can have only one entry point script and the entry point script must be only one script type. Your entry point script can, however, load multiple custom modules as dependencies. There is no limit to the number of dependencies your entry point script can load.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Global object</th>
</tr>
</thead>
<tbody>
<tr>
<td>define Object</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>
### 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>id</td>
<td>string</td>
<td>optional</td>
<td>Defines the id of the module</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>dependencies</td>
<td>string []</td>
<td>optional</td>
<td>Represents any module dependencies required by the callback function. Use the following syntax:</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Native SuiteScript 2.0 modules: [‘N/&lt;module name&gt;’]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Custom modules: [/&lt;path to module file in File Cabinet&gt;/&lt;module name&gt;] For other options, see the help topic Module Dependency Paths.</td>
<td></td>
</tr>
<tr>
<td>moduleObject</td>
<td>Function</td>
<td>required</td>
<td>A callback function or a module object</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODULE DOES NOT_EXIST</td>
<td>Module does not exist: {module path/name}</td>
<td>The NetSuite module or custom module dependency does not exist. If multiple modules do not exist, NetSuite only reports the first error encountered. If you receive this error, verify that all module paths and names are correct.</td>
</tr>
</tbody>
</table>

### Syntax for Module ID

The following code snippet shows sample syntax for the define(id, [dependencies,] callback) function signature. It is not a functional example or complete list.

```javascript
... 
define('mymodule', ['/test', '/sample'], function(test, sample){...});
...
```

### Syntax for Entry Point Script

The following code snippet shows a sample SuiteScript user event script type that creates a Phone Call record on the afterSubmit trigger.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType UserEventScript
 */

define(['N/record'],
    function (record)
    {
        function createPhoneCall(context)
```
```javascript
{
    if (context.type !== context.UserEventTypes.CREATE)
        return;
    var customerRecord = context.newRecord;
    if (customerRecord.getValue('salesrep')) {
        var call = record.create({
            type: record.Type.PHONE_CALL,
            isDynamic: true
        });
        call.setValue({
            fieldId: 'title',
            value: 'Make follow-up call to new customer'
        });
        call.setValue('assigned', customerRecord.getValue('salesrep'));
        call.setValue('phone', customerRecord.getValue('phone'));
        try {
            var callId = call.save();
            log.debug({
                title: 'Call record created successfully',
                details: 'Id: ' + callId
            });
        } catch (e) {
            log.error(e.name);
        }
    }
    return {
        afterSubmit: createPhoneCall
    };
}
```

Syntax for Custom Module

The following code snippets show the syntax for creating a custom SuiteScript 2.0 module in the script file `lib.js`.

```javascript
// lib.js
define(['./api/bar'], function(bar){     // require bar custom module
    return {
        makeSomething: function(){            // define function lib.makeSomething()
            var barObj = bar.create();   // use create() function from bar custom module
            return bar.convertToThing();   // returns the value of bar module function convertToThing()
        }
    };
});
```

The following code snippet shows the syntax for calling the function `lib` from the custom module `test.js` in a separate script file:

```javascript
// test.js
require(['lib'], function (lib) {    // require custom module (defined above)
```
require Function

The require Function is a global object that implements the require() Module Loader interface for SuiteScript 2.0. It conforms to the Asynchronous Module Definition (AMD) specification. When NetSuite executes the require() Function, it executes the callback function and loads the dependencies when they are needed.

This function executes asynchronously on the client side and synchronously on the server side.

**Note:** Only use the require() Function if you want to loading an existing module. If you want to create an entry point script or a new custom module, use the define Object.

Use the require() Function to achieve progressive loading of native SuiteScript 2.0 modules and custom modules. When you use the require() Function, dependencies are not loaded until they are needed. This can help increase script performance.

For example, if you add `lib1` as a dependency. When you call a method that is part of `lib1`, the Module Loader loads the module and executes the method. See Syntax.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td><code>require([dependencies,] callback)</code></td>
<td>Void</td>
<td>Loads a SuiteScript 2.0 entry point script or a SuiteScript 2.0 custom module.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Executes the callback function and loads the dependencies when they are required.</td>
</tr>
</tbody>
</table>

**Note:** To configure a require Object, you can associate a script to a JSON configuration file using a JSDoc tag. This is helpful to configure loading of a custom module. Properties that can hold feature metadata, aliases, paths, package, and mapping information related to a module id are supported. See require Configuration.

require([dependencies,] callback)

<table>
<thead>
<tr>
<th>Method Description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Function used to load a module only when the module is needed. When NetSuite executes the require() Function, it executes the callback function and loads the dependencies when they are required.</td>
<td></td>
</tr>
<tr>
<td>If you add a module as a dependency and the module is never used, the dependency is never loaded.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** This function conforms to the Asynchronous Module Definition (AMD) specification. For more information, see require Function.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Void</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global object</td>
<td><code>require Function</code></td>
</tr>
<tr>
<td>Since</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>
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>dependencies</td>
<td>string []</td>
<td>Optional</td>
<td>Represents any module dependencies required by the callback function.</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the following syntax:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ Native SuiteScript 2.0 modules: ['N/&lt;module name&gt;']</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ Custom modules: ['/&lt;path to module file in File Cabinet&gt;/&lt;module name&gt;']</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See the help topic Module Dependency Paths.</td>
<td></td>
</tr>
<tr>
<td>callback</td>
<td>Function</td>
<td>Required</td>
<td>Callback function to execute. Dependent modules are not loaded until they are required.</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODULE_DOES_NOT_EXIST</td>
<td>Module does not exist: {module path/name}</td>
<td>The NetSuite module or custom module dependency does not exist.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If multiple modules do not exist, NetSuite only reports the first error encountered. If you receive this error, verify that all module paths and names are correct.</td>
</tr>
</tbody>
</table>

Syntax

The following example shows progressive loading of modules in a script. For a functional example, see require Function.

```javascript
define({
    newInstance: function (type)
    {
        switch (type)
        {
            case 'lib1':
                require(['/lib1'], function (lib1) // Module Loader loads lib1
                {
                    return new lib1();
                })
                break;
            case 'lib2':
                require(['/lib2'], function (lib2) // Module Loader loads lib2
                {
                    return new lib2();
                })
                break;
            default:
                return null;
        }
    }
})
```
require Configuration

SuiteScript provides advanced options that provide you with greater control over require configuration.

If you set up a valid @NAmConfig JSDoc tag, SuiteScript implements the require configuration settings before loading dependencies. Configure the require Object before loading dependences so that you can run multiple client scripts with different configurations. Using the JSDoc tag can also support re-use by letting you use a common configuration across multiple scripts.

To configure a require Object, do the following:

- Add the @NAmConfig tag and provide a file cabinet path to the configuration file

```javascript
/**
 * @NAmConfig /SuiteScripts/configuration.json
 */
```

- SuiteScript will require a custom entry point module and its dependencies using the AMD configuration. For a list of supported configuration parameters, see require Configuration Parameters. Your require configuration must be in JSON format. For example:

```json
{
    "baseUrl" : "*/SuiteBundles"
}
```

⚠️ **Important:** Ensure that configuration file uses JSON syntax (and not JavaScript syntax). For more information about JSON, visit [http://json.org/](http://json.org/).

You can use `JSON.stringify(obj)` to convert a JavaScript object value to a key-value pair string in JSON form.

require Configuration Parameters


You can use the JSDoc tag to point a configuration file that holds the configuration values, such as when you want to set properties before loading a custom module, or set up configuration for improved compatibility.

The following configuration parameters are supported for require Object configuration:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Sample Usage</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseUrl</td>
<td>string</td>
<td>Optional</td>
<td>To configure a shorter relative path by indicating the root folder that holds the modules in the file cabinet.</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>paths</td>
<td>Object</td>
<td>Optional</td>
<td>To create a named alias to a path. For testing purposes, pass in an object that serves as a mock-up of another module.</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>
**require Function**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Sample Usage</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>packages</td>
<td>Object[]</td>
<td>Optional</td>
<td>To set up a custom name for a SuiteScript module</td>
<td></td>
</tr>
<tr>
<td>map</td>
<td>Object</td>
<td>Optional</td>
<td>To configure a special lookup suitable for traditional CommonJS packages that you want to use as a custom module.</td>
<td>Version 2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To specify an alias.</td>
<td>Release 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To handle multiple names for a module</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To load a set of identically-named but unique modules, such as dependency on multiple module versions.</td>
<td></td>
</tr>
<tr>
<td>config</td>
<td>Object</td>
<td>Optional</td>
<td>To assign attributes, such as metadata.</td>
<td>Version 2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Release 2</td>
</tr>
<tr>
<td>shim</td>
<td>Object</td>
<td>Optional</td>
<td>To prepare a non-AMD JS library for loading.</td>
<td>Version 2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Release 2</td>
</tr>
</tbody>
</table>

**require.config()**

Configuration of a require Object is optional and for advanced usage only. If you must configure a require Object, the @NamdConfig tag is suited for general use and is the preferred way to configure a require Object. However, existing scripts with calls to require.config can use this method with a context argument (although not recommended). Ensure that the call includes a context parameter and that its value is not a file path.

**log Object**

The log Object is loaded by default by NetSuite for all script types. You do not need to load it manually. However, you can choose to load it via N/log Module, such as for testing purposes.

**log Object Members**

- log.debug(options)
- log.audit(options)
- log.emergency(options)
- log.error(options)

For more details about the log Object and its methods, see N/log Module.

**util Object**

The util Object is loaded by default by NetSuite for all script types. You do not need to load it manually. However, you can choose to load it via N/util Module, such as for testing purposes.

**util Object Members**

- util.isArray(obj)
util Object

- `util.isBoolean(obj)`
- `util.isDate(obj)`
- `util.isFunction(obj)`
- `util.isNumber(obj)`
- `util.isObject(obj)`
- `util.isRegExp(obj)`
- `util.isString(obj)`

The `util` object also includes the following utility methods:

- `util.each(iterable, callback)`
- `util.extend(receiver, contributor)`
- `util.nanoTime()`

For more details about the `util` object and its methods, see N/util Module.

**toString()**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to determine an object's type. This is a global method that is loaded by default for all native SuiteScript 2.0 API objects.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Consider this method a replacement for the <code>instanceOf</code> operator (which is not supported). SuiteScript 2.0 members are immutable; you cannot construct or modify a native SuiteScript 2.0 member. Consequently, if you attempt to call <code>instanceOf</code>, an <code>undefined</code> error is thrown.</td>
</tr>
<tr>
<td><strong>Returns</strong></td>
<td>The object type as a string</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>

**Syntax**

The following code snippet shows the syntax for this member. It is not a functional example.

```javascript
... var type = mapContext.toString(); // When called on mapReduce.MapContext, toString returns "mapReduce.MapContext"
...
```

**JSON object**

SuiteScript 2.0 supports the JavaScript Object Notation (JSON) standard. You can use the JSON object to parse text as a JSON object and convert strings to JSON notation. For more information, see `JSON.parse(text)` and `JSON.stringify(obj)`. 
Important: The following sections are included as a summary and are intended for reference only. For additional information about JSON, see http://www.ietf.org/rfc/rfc4627.txt.

JSON.parse(text)

Method Description
Parse a string as a JSON object and returns the object.
The text parameter must conform to the JSON standard. See http://www.ietf.org/rfc/rfc4627.txt.

Returns
Object

Supported Script Types
All script types

Governance
None

Global object
JSON object

Since
Version 2015 Release 2

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>text</td>
<td>string</td>
<td>Required</td>
<td>Text to parse as a JSON object. The string must conform to the JSON standard.</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>reviver</td>
<td>Function</td>
<td>Optional</td>
<td>Specifies how to transform the parsed value before it is returned</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>

Syntax

The following code snippet shows the syntax for this member. It is not a functional example.

```javascript
... var text = '{ "employees" : [" +
  '{ "firstName":"John" ,  "lastName":"Doe" }, +
  '{ "firstName":"Anna" ,  "lastName":"Smith" }, +
  '{ "firstName":"Peter" ,  "lastName":"Jones" } ]}';
var obj = JSON.parse(text);
var firstEmp = obj.employees[1].firstName + " " + obj.employees[1].lastName;
...```

JSON.stringify(obj)

Method Description
Converts a JavaScript object or value to a JSON string
For more information about JSON object format, see http://www.ietf.org/rfc/rfc4627.txt.

Returns
JSON string

Supported Script Types
All script types
Governance | None
Global object | JSON object
Since | Version 2015 Release 2

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>value</td>
<td>Object</td>
<td>Required</td>
<td>The value to convert to a JSON string</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>replacer</td>
<td>Function</td>
<td>Optional</td>
<td>Function that changes the behavior of the stringification process</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>space</td>
<td>Object</td>
<td>Optional</td>
<td>A string or number that is used to insert white space in the output JSON string for readability</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>

Syntax

The following code snippet shows the syntax for this member. It is not a functional example.

```javascript
var contact = {
    firstName: 'John',
    lastName : 'Doe',
    jobTitle : 'CEO'
};

var jsonString = JSON.stringify(contact);
```

This method converts the `contact` object to the following string:

```javascript
{"firstName":"John","lastName":"Doe","jobTitle":"CEO"}
```

Promise Object

In SuiteScript 2.0, all client scripts support the use of Promises. With Promises, developers can write asynchronous code that is intuitive and efficient. SuiteScript 2.0 provides promise APIs for selected modules (see SuiteScript 2.0 Promise APIs). In addition, you can create custom Promises in all client scripts (see Custom Promises).

A promise is a JavaScript object that represents the eventual result of an asynchronous process. After this object is created, it serves as a placeholder for the future success or failure of the operation. During the period of time that the promise object is waiting, the remaining segments of the script can execute.

A Promise holds one of the following values:

- fulfilled - The operation is successful.
- rejected - The operation failed.
- pending - The operation is still in progress and has not yet been fulfilled or rejected.
When it is first created, a Promise holds the value `pending`. After the associated process is complete (from success or failure), the value changes to `fulfilled` or `rejected`. A success or failure callback function attached to the Promise is called when the process is complete. Note that a Promise can only succeed or fail one time. When the value of the Promise updates to `fulfilled` or `rejected`, it cannot change.

For additional information regarding Promises, see [https://www.promisejs.org/](https://www.promisejs.org/).

**SuiteScript 2.0 Promise APIs**

SuiteScript 2.0 provides client-side promise APIs. For supported modules members and additional API information, see [SuiteScript 2.0 Modules](#).

⚠️ **Important:** Although these modules as a whole are supported in client and server-side scripts, their promise APIs are supported only in client scripts.

The available promise APIs are named so that they correspond with their synchronous counterparts. The distinction is that the promise APIs have names that are suffixed with `.promise`. For example, the `search.create(options)` API has a promise version named `search.create.promise(options)`.

The following is a basic example of how to use a promise API in a client script.

```javascript
/**
 * @NAPIVersion 2.0
 */
define(['N/search'],
  function(search)
  {
    function doSomething()
    {
      search.create.promise({
        type: 'salesorder'
      }).then(function(result) {
        log.debug("Completed: " + result);
        //do something after completion
      }).catch(function(reason) {
        log.debug("Failed: " + reason)
        //do something on failure
      });
    }
    return {
      pageInit: doSomething
    }
  }
);
```

This example demonstrates how to chain promises created with promise APIs.

```javascript
/**
 * @NAPIVersion 2.0
 */
```
define(['N/search'],
    function(search)
    {
        function doSomething()
        {
            var filter = search.createFilter({
                name: 'mainline',
                operator: search.Operator.IS,
                values: ['T']
            });
            search.create.promise({
                type: 'salesorder',
                filters: [filter]
            }).then(function(searchObj) {
                return searchObj.run().each.promise{
                    function(result, index){
                        //do something
                    }
                }).then(function(result) {
                    log.debug("Completed: " + result)
                    //do something after completion
                }).catch(function(reason) {
                    log.debug("Failed: " + reason)
                    //do something on failure
                });
            })
            return
            {
                pageInit: doSomething
            }
        }
    });

Custom Promises

The following example shows a custom Promise. Custom Promises do not utilize the SuiteScript 2.0 promise APIs.

```javascript
/**
 * @NAPIVersion 2.0
 */
define(function(){
    function doSomething(addresses){
        var promise = new Promise(function(resolve, reject){
            var url = 'https://your.favorite.maps/api/directions?start=' + addresses.start + '&end=' + addresses.end,
                isAsync = true,
                xhr = new XMLHttpRequest();

            xhr.addEventListener('load', function (event) {
                if (xhr.readyState === 4) {
                    if (xhr.status === 200) {
```
resolve(xhr.responseText);
} else {
    reject(xhr.statusText);
}
}
)xhr.addEventListener('error', function (event) {
    reject(xhr.statusText);
});
xhr.open('GET', url, isAsync);
xhr.send();
}

return promise;
}

return {
    lookupDirections: doSomething
};
}
SuiteScript 2.0 Modules

SuiteScript 2.0 APIs are organized into various modules, based on behavior. These modules are described below.

**Note:** As a best practice, you should load only the modules that are needed by your script. However, you can load all SuiteScript 2.0 modules at one time. Do this by passing the modules' parent directory to the `define()` statement and its callback function: `define(['N'], function(N) {...});`. This is a convenient way to load all modules, but does sacrifice the performance advantage of loading only the modules that are needed. We provide this feature so that you can test and familiarize yourself with SuiteScript 2.0. We do not recommend that you load all modules at once in a production environment.

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/action Module</td>
<td>Load the N/action module APIs to execute business logic to update the state of a record. Action APIs emulate NetSuite UI buttons.</td>
</tr>
<tr>
<td>N/auth Module</td>
<td>Load the auth module when you want to change your NetSuite login credentials.</td>
</tr>
<tr>
<td>N/cache Module</td>
<td>Load the cache module to enable the caching of needed data and improve performance.</td>
</tr>
<tr>
<td>N/config Module</td>
<td>Load the config module when you want to access NetSuite configuration settings. See <code>config.Type</code> for a list of supported configuration pages.</td>
</tr>
<tr>
<td>N/crypto Module</td>
<td>Load the crypto module to work with hashing, hash-based message authentication (hmac), and symmetrical encryption. Access wrappers for OpenSSL's hash, hmac, cipher, and decipher methods.</td>
</tr>
<tr>
<td>N/currency Module</td>
<td>Load the currency module to work with exchange rates within your NetSuite account. You can use the currency module to find the exchange rate between two currencies based on a certain date.</td>
</tr>
<tr>
<td>N/currentRecord Module</td>
<td>Load the currentRecord module to access the record instance that you are currently working on. You can then use the record instance in a client-side context.</td>
</tr>
<tr>
<td>N/email Module</td>
<td>Load the email module when you want to send email messages from within NetSuite. You can use the email module to send regular, bulk, and campaign email.</td>
</tr>
<tr>
<td>N/encode Module</td>
<td>Load the encode module when you want to convert a string to another type of encoding. See <code>encode.Encoding</code> for a list of supported character set encoding.</td>
</tr>
<tr>
<td>N/error Module</td>
<td>Load the error module when you want to create your own custom SuiteScript errors. Use these custom errors in try-catch statements to abort script execution.</td>
</tr>
<tr>
<td>N/file Module</td>
<td>Load the file module to work with files in NetSuite.</td>
</tr>
<tr>
<td>N/format Module</td>
<td>Load the format module to convert strings into a specified format and to parse formatted data into strings.</td>
</tr>
<tr>
<td>N/format/i18n Module</td>
<td>Load the format/i18n module to format currency.</td>
</tr>
<tr>
<td>N/http Module</td>
<td>Load the http module to make http calls. All HTTP content types are supported.</td>
</tr>
<tr>
<td>N/https Module</td>
<td>Load the https module to make https calls. You can also use this module to encode binary content or securely access a handle to the value in a NetSuite credential field.</td>
</tr>
<tr>
<td>N/log Module</td>
<td>Load the log module when you want to access methods for logging script execution details. Module members are also supported by the global <code>log Object</code>.</td>
</tr>
<tr>
<td>Module</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>N/plugin Module</td>
<td>Load the plugin module to load custom plug-in implementations.</td>
</tr>
<tr>
<td>N/portlet Module</td>
<td>Load the portlet module when you want to resize or refresh a form portlet.</td>
</tr>
<tr>
<td>N/query Module</td>
<td>Load the query module to create and run searches using the SuiteAnalytics Workbook query engine.</td>
</tr>
<tr>
<td>N/record Module</td>
<td>Load the record module to work with NetSuite records.</td>
</tr>
</tbody>
</table>
| N/redirect Module     | Load the redirect module when you want to redirect users to one of the following:  
  - URL  
  - Suitelet  
  - Record  
  - Task link  
  - Saved search  
  - Unsaved search |
| N/render Module       | Load the render module to create forms or email from templates and to print to PDF or HTML. |
| N/runtime Module      | Load the runtime module when you want to access the runtime settings for company, script, session, system, user, or version. |
| N/search Module       | Load the search module to create and run on demand or saved searches and analyze and iterate through the search results. You can search for a single record by keywords, create saved searches, search for duplicate records, or return a set of records that match filters you define. |
| N/sftp Module         | Load the sftp module to connect to a remote FTP server via SFTP and transfer files. |
| N/sso Module          | Load the sso module when you want to generate outbound single sign-on (SuiteSignOn) tokens. |
| N/task Module         | Load the task module to create tasks and place them in the internal NetSuite scheduling or task queue. Use the task module to schedule scripts, run Map/Reduce scripts, import CSV files, merge duplicate records, and execute asynchronous workflows. |
| N/transaction Module  | Load the transaction module to void transactions.                          |
| N/translation Module  | Load the translations module to load NetSuite Translation Collections in SuiteScript. |
| N/ui/dialog Module    | Load the dialog module to load a modal dialog that persists until a button on the dialog is pressed. |
| N/ui/message module   | Load the message module to display a message at the top of the screen under the menu bar. |
| N/ui/serverWidget Module | Load the serverWidget module when you want to work with the user interface within NetSuite. |
| N/url Module          | Load the url module when you want to determine URL navigation paths within NetSuite or format URL strings. |
| N/util Module         | Load the util module when you want to manually access util methods. Module members are also supported by the global util Object. |
| N/workflow Module     | Load the workflow module to initiate new workflow instances or trigger existing workflow instances. |
The N/action module APIs let you execute business logic to update the state of records in view mode. To execute business logic on records that you are editing, use the record macro APIs, which are included in the N/record Module module. See Record Object Members and Macro Object Members. Action and Macro APIs are the programmatic equivalent to clicking a button in the UI. To learn more, see the help topic Overview of Record Action and Macro APIs.

The changes that you make to records with N/action module APIs are persisted in the database immediately. For example, consider the timebill record. After you click the Approve button in the UI, the timebill and its entries are saved in an approved state, and this change is immediately updated in the database.

Governance for action module APIs varies for actions and record types. See the action help for governance information specific to actions and record types.

A limited number of individual actions for specific record types are supported. For details, see the help topic Supported Record Actions.

N/action Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>action.Action</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a NetSuite record action.</td>
</tr>
<tr>
<td>Plain JavaScript Object</td>
<td>action.execute(options)</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>A plain JavaScript object of actions available for a record type.</td>
</tr>
<tr>
<td>Method</td>
<td>action.execute(Beta)</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Executes the record action and returns action results in an object.</td>
</tr>
<tr>
<td></td>
<td>action.executeBulk(Beta)</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Asynchronously executes the record action and returns the action results in an object.</td>
</tr>
<tr>
<td></td>
<td>action.find(options)</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Returns a plain JavaScript object of available record.</td>
</tr>
</tbody>
</table>
### Action Object Members

The following members are called on `action.Action`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>Action(options)</code></td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Executes the action and returns the action results in an object.</td>
</tr>
<tr>
<td></td>
<td><code>Action.promise(options)</code></td>
<td>Promise</td>
<td>Client scripts</td>
<td>Executes the action asynchronously and returns the action results in an object.</td>
</tr>
<tr>
<td></td>
<td><code>Action.execute(options)</code></td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Executes the action and returns the action results in an object.</td>
</tr>
<tr>
<td></td>
<td><code>Action.execute().promise()</code></td>
<td>Promise</td>
<td>Client scripts</td>
<td>Executes the action asynchronously and returns the action results in an object.</td>
</tr>
<tr>
<td></td>
<td><code>Action.executeBulk()</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Executes an asynchronous bulk record action and returns its task ID for later status inquiry.</td>
</tr>
<tr>
<td></td>
<td><code>Action.getBulkStatus()</code></td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Returns the current status of <code>action.executeBulk(options)</code> (Beta) with the given task ID.</td>
</tr>
<tr>
<td>Property</td>
<td><code>Action.description</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The action description.</td>
</tr>
<tr>
<td></td>
<td><code>Action.id</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The ID of the action. For a list of action IDs, see the help topic Supported Record Actions.</td>
</tr>
<tr>
<td></td>
<td><code>Action.label</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The action label.</td>
</tr>
<tr>
<td></td>
<td><code>Action.parameters</code></td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>The action parameters.</td>
</tr>
<tr>
<td></td>
<td><code>Action.recordType</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The type of the record on which the action is to be performed.</td>
</tr>
</tbody>
</table>
N/action Module Script Samples

These samples use the `require` function, so that you can copy each script into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For more information, see SuiteScript 2.0 Global Objects and SuiteScript 2.0 Script Types.

**Important:** The samples included in this section are intended to show how actions work in SuiteScript at a high-level. For specific samples, see the help topic Supported Record Actions.

The following server script sample finds and executes an action on the timebill record without promises.

```javascript
require(['N/action', 'N/record'], function(action, record) {
  // create timebill record
  var rec = record.create({type: 'timebill', isDynamic: true});
  rec.setValue({fieldId: 'employee', value: 104});
  rec.setValue({fieldId: 'location', value: 312});
  rec.setValue({fieldId: 'hours', value: 5});
  var recordId = rec.save();

  var actions = action.find({
    recordType: 'timebill',
    recordId: recordId
  });

  log.debug("We've got the following actions: " + Object.keys(actions));
  if (actions.approve) {
    var result = actions.approve();
    log.debug("Timebill has been successfully approved");
  } else {
    log.debug("The timebill is already approved");
  }
});
```

// Outputs the following:
// We've got the following actions: approve, reject
// Timebill has been successfully approved

The following client-side script sample asynchronously finds actions available for a timebill record and then executes one with promises.

```javascript
require(['N/action', 'N/record'], function(action, record) {
  // create timebill record
  var rec = record.create({type: 'timebill', isDynamic: true});
  rec.setValue({fieldId: 'employee', value: 104});
  rec.setValue({fieldId: 'location', value: 312});
  rec.setValue({fieldId: 'hours', value: 5});
```
var recordId = rec.save();

// find all qualified actions and then execute approve if available
action.find.promise(
    recordType: 'timebill',
    recordId: recordId
).then(function(actions) {
    console.log("We've got the following actions: " + Object.keys(actions));
    if (actions.approve) {
        actions.approve.promise().then(function(result) {
            console.log("Timebill has been successfully approved");
        });
    } else {
        console.log("The timebill is already approved");
    }
});

// Outputs the following:
// We've got the following actions:
// The timebill has been successfully approved

The following sample uses action.executeBulk(options).

require(['N/action', 'N/util']) function(action, util) {

    // 1a) Bulk execute the specified action on a provided list of record IDs.
    // The params property is an array of parameter objects where each object contains mandatory recordId and arbitrary additional parameters.
    var handle = action.executeBulk({
        recordType: 'timebill',
        id: "approve",
        params: [{ recordId: 1, note: "this is a note for 1" },
                  { recordId: 5, note: "this is a note for 5" },
                  { recordId: 23, note: "this is a note for 23" }]
    });

    // 1b) Bulk execute the specified action on a provided list of record IDs.
    // The parameters in the previous example are very similar and can be generated programatically using the map function.
    var searchResults = /* result of a search, e.g. [1, 5, 23] */;
    var handle = action.executeBulk({
        recordType: 'timebill',
        id: "approve",
        params: searchResults.map(function(v) {
            return { recordId: v, note: "this is a note for " + v };});
    });

    // 2a) Bulk execute the specified action on a provided list of record IDs.
    // This time with homogenous parameters, i.e. all parameter objects are equal except recordId.
    var handle = action.executeBulk({
        recordType: 'timebill',
        id: "approve",
        params: [{ recordId: 1, note: "this is a note for 1" },
                  { recordId: 5, note: "this is a note for 5" },
                  { recordId: 23, note: "this is a note for 23" }]
    });
}
// 2b) Bulk execute the specified action on a provided list of record IDs.
// This time with homogenous parameters. Equivalent to the previous example.
var commonParams = {foo: "bar", name: "John Doe"};
var handle = action.executeBulk({
  recordType: "timebill",
  id: "approve",
  params: searchResults.map(function(v) {
    return util.extend({recordId: v, commonParams});
  })
});

// 3) Bulk execute the specified action on a provided list of record IDs.
// This is the simplest usage with no extra parameters besides the record ID.
var handle = action.executeBulk({
  recordType: "timebill",
  id: "approve",
  params: searchResults.map(function(v) {return {recordId: v}})
});

// 4) Bulk execute the specified action on all record instances that qualify.
// Since we don’t have a list of record IDs in hand, we only provide the callback
// that will later be used to transform a record ID to the corresponding parameters object.
var handle = action.executeBulk({
  recordType: "timebill",
  id: "approve",
  condition: action.ALL_QUALIFIED_INSTANCES,
  paramCallback: function(v) {
    return { recordId: v, note: "this is a note for " + v }
  }
});

// 5) Get a particular action for a particular record type.
var approveTimebill = action.get({
  recordType: "timebill",
  id: "approve"
});

// 6) Bulk execute the previously obtained action on a provided list of record IDs.
// Params are generated the same way as above in action.executeBulk().
var handle = approveTimebill.executeBulk({
  params: searchResults.map(function(v) {
    return { recordId: v, note: "this is a note for " + v }
  })
});

// 7) Bulk execute the previously obtained action on all record instances that qualify.
var handle = approveTimebill.executeBulk({
  condition: action.ALL_QUALIFIED_INSTANCES,
  paramCallback: function(v) {
    return { recordId: v, note: "this is a note for " + v }
  }
});
action.Action

Object Description
Encapsulates a NetSuite record action.

This object is returned by the action.get(options) and action.find(options) methods.

Supported Script Types
Client and server scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Module
N/action Module

Methods and Properties
Action Object Members

Since
2018.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code
...
var action = actionMod.get({recordType: 'timebill', id: 'approve'});
...
// Add additional code
```

Action(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Executes the action and returns the action result in a plain JavaScript object.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The action result is returned in an object. The response property of the results object shows the action result. If the action fails, it is listed in the results object's notifications property. If the action executes successfully, the notifications property is usually empty.</td>
</tr>
<tr>
<td></td>
<td>If the Action object is qualified (it is a result of an action.get() or action.find() call that provides the recordId), then it is not required to provide a recordId and the options.params.recordId parameter is optional. If options.params.recordId is provided during execution, it takes precedence over the recordId stored in the Action object.</td>
</tr>
</tbody>
</table>

⚠️ Note: Replace Action with the name of the action you are executing.

Returns
Object
Supported Script Types

Client and server scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Module
N/action Module

Parent Object
action.Action

Sibling Object Members
Action Object Members

Since
2018.2

Parameters

Note: The parameters that are required vary for action types. The only parameter that is always required is options.recordid, unless the action object is qualified. An action object is qualified if it is the result of an action.get() or action.find() call that provides the recordId.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.Object</td>
<td>Object</td>
<td>required or optional</td>
<td>The parameters that need to be provided depend on the action implementation. See the action help.</td>
</tr>
<tr>
<td>options.params.recordId</td>
<td>string</td>
<td>required or optional</td>
<td>The record instance ID of the record on which the action is to be performed. This is the NetSuite record internal ID.</td>
</tr>
</tbody>
</table>

Errors

Syntax

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
</tbody>
</table>

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code
...
var result = action({recordId: 1});
...
// Add additional code
```

Action.promise(options)

Method Description
Executes the action asynchronously and returns the action result in a plain JavaScript object. The action result is returned in an object. The response property of the results object shows the action result. If the action fails, it is listed in the results object's notifications property. If the action executes successfully, the notifications property is usually empty.
If the Action object is qualified (it is a result of an action.get() or action.find() call that provides the recordId), then it is not required to provide a recordId and the options.params.recordId parameter is optional. If options.params.recordId is provided during execution, it takes precedence over the recordId stored in the Action object.

**Note:** Replace Action with the name of the action you are executing.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Promise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td></td>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/action Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>action.Action</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Action Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The parameters that are required vary for action types. The only parameter that is always required is options.recordId, unless the action object is qualified. An action object is qualified if it is the result of an action.get() or action.find() call that provides the recordId.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.Object</td>
<td>Object</td>
<td>required or optional</td>
<td>The parameters that need to be provided depend on the action implementation. See the action help.</td>
</tr>
<tr>
<td>options.params.recordId</td>
<td>string</td>
<td>required or optional</td>
<td>The record instance ID of the record on which the action is to be performed. This is the NetSuite record internal ID.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code
...
action.promise({recordId: 1}).then(function(result) { /* process result here */ });
...
// Add additional code
```
Action.execute(options)

**Method Description**

Executes the action and returns the action result in a object.

The `response` property of the result object holds the actual response returned by the action implementation. The `notifications` property of the result object is an array of notification objects. It contains the details of errors and warnings that occurred during action execution. If the action executes successfully, the `notifications` property is usually empty.

If the Action object is qualified (it is a result of an action.get() or action.find() call that provides the recordId), then it is not required to provide a recordId and the `options.params.recordId` parameter is optional. If `options.params.recordId` is provided during execution, it takes precedence over the `recordId` stored in the Action object.

**Returns**

Object

**Supported Script Types**

Client and server scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](https://www.netsuite.com/ncrs/2.0/).

**Module**

N/action Module

**Parent Object**

action.Action

**Sibling Object Members**

Action Object Members

**Since**

2018.2

**Parameters**

**Note:** The parameters that are required vary for action types. The only parameter that is always required is `options.recordId`, unless the action object is qualified. An action object is qualified if it is the result of an action.get() or action.find() call that provides the recordId.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.Object</td>
<td>Object</td>
<td>required or optional</td>
<td>The parameters that need to be provided depend on the action implementation. See the action help.</td>
</tr>
<tr>
<td>options.params.recordId</td>
<td>string</td>
<td>required or optional</td>
<td>The record instance ID of the record on which the action is to be performed. This is the NetSuite record internal ID.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/action Module Script Samples](https://www.netsuite.com/ncrs/2.0/).

```javascript
// Add additional code
```
var result = action.execute({recordId: 1});

// Add additional code

### Action.execute.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Executes the action asynchronously and returns the action result in a plain JavaScript object. The action result is returned in an object. The <code>response</code> property of the results object shows the action result. If the action fails, it is listed in the results object's <code>notifications</code> property. If the action executes successfully, the <code>notifications</code> property is usually empty. If the Action object is qualified (it is a result of an action.get() or action.find() call that provides the recordId), then it is not required to provide a <code>recordId</code> and the <code>options.params.recordId</code> parameter is optional. If <code>options.params.recordId</code> is provided during execution, it takes precedence over the <code>recordId</code> stored in the Action object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Promise</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/action Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>action.Action</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Action Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The parameters that are required vary for action types. The only parameter that is always required is `options.recordId`, unless the action object is qualified. An action object is qualified if it is the result of an action.get() or action.find() call that provides the recordId.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.Object</code></td>
<td>Object</td>
<td>required or optional</td>
<td>The parameters that need to be provided depend on the action implementation. See the action help.</td>
</tr>
<tr>
<td><code>options.params.recordId</code></td>
<td>string</td>
<td>required or optional</td>
<td>The record instance ID of the record on which the action is to be performed. This is the NetSuite record internal ID.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
</tbody>
</table>
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code
...
action.execute.promise({recordId: 1}).then(function(result) { /* process result here */ });
...
// Add additional code
```

### Action.executeBulk(options) (Beta)

**Warning:** The N/action module bulk action APIs are a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature's operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

#### Method Description
Executes an asynchronous bulk record action and returns its task ID for status queries with action.getBulkStatus(options) (Beta).

#### Returns
- **string**

#### Supported Script Types
- Client and server-side scripts
  - For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

#### Governance
- 50 usage units

#### Module
- N/action Module

#### Parent Object
- action.Action

#### Sibling Object Members
- Action Object Members

#### Since
- 2019.1

### Parameters

**Note:** The options.params array consists of parameter objects. The values that are required in each parameter object vary for action types. The only value that is always required is options.recordId, unless the action object is qualified. An action object is qualified if it is the result of an action.get() or action.find() call that provides the recordId.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.params</td>
<td>array</td>
<td>optional</td>
<td>The options.params parameter is mutually exclusive to options.condition and options.paramCallback. An array of parameter objects. Each object corresponds to one record ID of the record for which the action is to be executed. The object has the following form:</td>
</tr>
</tbody>
</table>
### Parameter | Type | Required / Optional | Description
--- | --- | --- | ---
| | | | {recordId: 1, someParam: 'example1', otherParam: 'example2'}

**options.condition** | string | optional | The condition used to select record IDs of records for which the action is to be executed. Only the `action.ALL_QUALIFIED_INSTANCES` constant is currently supported.

The `action.ALL_QUALIFIED_INSTANCES` condition only works correctly if the author of the record action has implemented the `findInstances` method of the `RecordActionQualifier` interface. An example of such action is `approve` on the timebill and timesheet records.

**options.paramCallback** | string | optional | the name of the function that takes a record ID and returns the parameter object for the specified record ID.

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type. – or – The action exists, but cannot be executed on the specified record instance.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code

var actionObj = action.get({
    recordType: 'timebill',
    id: 'approve'
});

var handle = actionObj.executeBulk({
    params: [
        {
            recordId: 1, note: 'this is a note for 1'
        },
        {
            recordId: 5, note: 'this is a note for 5'
        },
        {
            recordId: 23, note: 'this is a note for 23'
        }
    ]
});
```
action.getBulkStatus(options) (Beta)

**Warning:** The N/action module bulk action APIs are a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature's operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the current status of action.executeBulk(options) (Beta) for the specified task ID. The bulk execution status is returned in a status object.</td>
<td>RecordActionTaskStatus (Beta) Object Members</td>
<td>Client and server-side scripts</td>
<td>N/action Module</td>
<td>N/action Module Members</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.taskId</td>
<td>string</td>
<td>required</td>
<td>The task ID returned by a previous action.executeBulk(options) (Beta) call.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code

// Obtain the status as a RecordActionTaskStatus object
...
var res = action.getBulkStatus({
  taskId: handle
});
// Add additional code
```

**Action.description**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The action description.</th>
</tr>
</thead>
</table>
**Action.id**

**Property Description**
The ID of the action.
For a list of action IDs, see the help topic [Supported Record Actions](#).

**Type**
string

**Supported Script Types**
Client and server scripts
For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/action Module

**Sibling Object Members**
Action Object Members

**Since**
2018.2

**Syntax**

```
// Add additional code
...
var id = action.id; // get the id of the action
...
// Add additional code
```
### Action.parameters

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The action parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server 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>

**Module**  
N/action Module

**Sibling Object Members**  
Action Object Members

**Since**  
2018.2

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/action Module Script Samples](#).

```javascript
// Add additional code
...
var params = action.parameters; // get the action parameters
...
// Add additional code
```

### Action.recordType

| Property Description | The type of the record on which the action is to be performed. |

**Module**  
N/action Module

**Sibling Object Members**  
Action Object Members

**Since**  
2018.2

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/action Module Script Samples](#).

```javascript
// Add additional code
...
var params = action.parameters; // get the action parameters
...
// Add additional code
```
For a list of record types, see `record.Type`.

**Type**

string

**Supported Script Types**

Client and server scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/action Module

**Sibling Object Members**

Action Object Members

**Since**

2018.2

---

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/action Module Script Samples](#).

```javascript
// Add additional code
...
var recordType = action.recordType; // get the record type
...
// Add additional code
```

#### action.execute(options)

**Method Description**

Executes the record action and returns the action results in a plain JavaScript object. If the action fails, it is listed in the results object's `notifications` property. If the action executes successfully, the `notifications` property is usually empty.

**Returns**

Object

**Supported Script Types**

Client and server scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/action Module

**Sibling Object Members**

N/action Module Members

**Since**

2018.2

---

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.recordType</code></td>
<td>string</td>
<td>required</td>
<td>The record type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a list of record types, see <code>record.Type</code>.</td>
</tr>
<tr>
<td><code>options.id</code></td>
<td>string</td>
<td>required</td>
<td>The action ID.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>---------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>required</td>
<td>Action arguments.</td>
</tr>
<tr>
<td>options.params.recordId</td>
<td>string</td>
<td>required</td>
<td>The record instance ID. This is the NetSuite record internal ID.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type.</td>
</tr>
<tr>
<td></td>
<td>- or -</td>
</tr>
<tr>
<td></td>
<td>The action exists, but cannot be executed on the specified record instance.</td>
</tr>
<tr>
<td>RECORD_DOES_NOT_EXIST</td>
<td>The specified record instance does not exist.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
// Add additional code
...
var result = actionMod.execute({id: 'note', recordType: 'timebill', params: {recordId:1}});
...
// Add additional code
```

**action.execute.promise(options)**

**Method Description**
Executes the record action asynchronously. If the action fails, it is listed in the results object's `notifications` property. If the action executes successfully, the `notifications` property is usually empty.

**Returns**
Promise

**Supported Script Types**
Client scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/action Module
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.recordType</td>
<td>string</td>
<td>required</td>
<td>The record type. For a list of record types, see record.Type.</td>
</tr>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The action ID. For a list of action IDs, see the help topic Supported Record Actions.</td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>required</td>
<td>Action arguments.</td>
</tr>
<tr>
<td>options.params.recordId</td>
<td>string</td>
<td>required</td>
<td>The record instance ID. This is the NetSuite record internal ID.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type.</td>
</tr>
<tr>
<td></td>
<td>- or -</td>
</tr>
<tr>
<td></td>
<td>The action exists, but cannot be executed on the specified record instance.</td>
</tr>
<tr>
<td>RECORD_DOES_NOT_EXIST</td>
<td>The specified record instance does not exist.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code
...
actionMod.execute.promise({id: 'note', recordType: 'timebill', params: {recordId: 1}}).then(function(result) {
  // do something with the result
});
...
// Add additional code
```
### action.executeBulk(options) (Beta)

**Warning:** The N/action module bulk action APIs are a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature's operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Executes an asynchronous bulk record action and returns its task ID for status queries with <code>action.getBulkStatus(options)</code> (Beta). The <code>options.params</code> parameter is mutually exclusive to <code>options.condition</code> and <code>options.paramCallback</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>50 usage units</td>
</tr>
<tr>
<td>Module</td>
<td>N/action Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>N/action Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The `options` parameter is a JavaScript object. The `options.params` array consists of parameter objects. The values that are required in each parameter object vary for action types. The only value that is always required is `recordId`.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.recordType</code></td>
<td>string</td>
<td>required</td>
<td>The record type. For a list of record types, see <code>record.Type</code>.</td>
</tr>
<tr>
<td><code>options.id</code></td>
<td>string</td>
<td>required</td>
<td>The action ID.</td>
</tr>
<tr>
<td><code>options.params</code></td>
<td>array</td>
<td>optional</td>
<td>An array of parameter objects. Each object corresponds to one record ID of the record for which the action is to be executed. The object has the following form:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(recordId: 1, someParam: 'example1', otherParam: 'example2')</td>
</tr>
<tr>
<td><code>options.condition</code></td>
<td>string</td>
<td>optional</td>
<td>The condition used to select record IDs of records for which the action is to be executed. Only the action.ALL_QUALIFIED_INSTANCES constant is currently supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The action.ALL_QUALIFIED_INSTANCES condition only works correctly if the author of the record action has implemented</td>
</tr>
</tbody>
</table>
the `findInstances` method of the `RecordActionQualifier` interface. An example of such action is approve on the timebill and timesheet records.

| options.paramCallback | string | optional | Function that takes record ID and returns the parameter object for the specified record ID. |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>The <code>options.recordType</code> parameter is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type.</td>
</tr>
<tr>
<td></td>
<td>- or -</td>
</tr>
<tr>
<td></td>
<td>The action exists, but cannot be executed on the specified record instance.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/action Module Script Samples](#).

```javascript
// Add additional code
...
var handle = action.executeBulk({
    recordType: 'timebill',
    id: 'approve',
    params: [
        { recordId: 1, note: 'this is a note for 1' },
        { recordId: 5, note: 'this is a note for 5' },
        { recordId: 23, note: 'this is a note for 23' }
    ]
});
// Add additional code
```

`action.find(options)`

**Method Description**

Performs a search for available record actions. If only the `recordType` parameter is specified, all actions available for the record type are returned. If the `recordId` parameter is also specified, then only actions that qualify for execution on the given record instance are returned. If the `id` parameter is specified, then only the action with the specified action ID is returned.

This method returns a plain JavaScript object of NetSuite record actions available for the record type. The object contains one or more `action.Action` objects. If there are no available actions for the specified record type, an empty object is returned.

If the `recordId` is specified in this call, the actions that are found are considered qualified. You do not have to provide the `recordId` to execute a qualified action.

**Returns**

Object
Supported Script Types
Client and server scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Module
N/action Module

Sibling Object Members
N/action Module Members

Since
2018.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.recordType</td>
<td>string</td>
<td>required</td>
<td>The record type. For a list of record types, see record.Type.</td>
</tr>
<tr>
<td>options.recordId</td>
<td>string</td>
<td>optional</td>
<td>The record instance ID.</td>
</tr>
<tr>
<td>options.id</td>
<td>string</td>
<td>optional</td>
<td>The action ID.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>The options.recordType parameter is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type.</td>
</tr>
<tr>
<td></td>
<td>- or - The action exists, but cannot be executed on the specified record instance.</td>
</tr>
<tr>
<td>RECORD_DOES_NOT_EXIST</td>
<td>The specified record ID does not exist.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```
// Add additional code
...
var actions = action.find({
  recordType: 'timebill',
  recordId: recordId
});
...
// Add additional code
```
action.find.promise(options)

**Method Description**
Performs a search for available record actions asynchronously. If only the `recordType` parameter is specified, all actions available for the record type are returned. If the `recordId` parameter is also specified, then only actions that qualify for execution on the given record instance are returned. If the `id` parameter is specified, the only the action with the specified action ID is returned.

This method returns a plain JavaScript object of NetSuite record actions available for the record type. The object contains one or more `action.Action` objects. If there are no available actions for the specified record type, an empty object is returned.

If the `recordId` is specified in this call, the actions that are found are considered qualified. You do not have to provide the `recordId` to execute a qualified action.

**Returns**
Promise

**Supported Script Types**
Client scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/action Module

**Sibling Object Members**
N/action Module Members

**Since**
2018.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.recordType</code></td>
<td>string</td>
<td>required</td>
<td>The record type. For a list of record types, see <code>record.Type</code>.</td>
</tr>
<tr>
<td><code>options.recordId</code></td>
<td>string</td>
<td>optional</td>
<td>The record instance ID.</td>
</tr>
<tr>
<td><code>options.id</code></td>
<td>string</td>
<td>optional</td>
<td>The action ID.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>The <code>options.recordType</code> parameter is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type. - or - The action exists, but cannot be executed on the specified record instance.</td>
</tr>
<tr>
<td>RECORD DOES NOT_EXIST</td>
<td>The specified record ID does not exist.</td>
</tr>
</tbody>
</table>
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code
...
var promise = action.find.promise({recordType: 'timebill'});
    promise.then(function(actionList) {
        // do something with the list of actions
    });
...
// Add additional code
```

### action.get(options)

**Method Description**

Returns an executable record action for the specified record type. If the `recordId` parameter is specified, the action object is returned only if the specified action can be executed on the specified record instance.

**Returns**

`action.Action`

**Supported Script Types**

Client and server scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

<table>
<thead>
<tr>
<th>Module</th>
<th>N/action Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sibling Object Members</td>
<td>N/action Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.recordType</td>
<td>string</td>
<td>required</td>
<td>The record type. For a list of record types, see <code>record.Type</code>.</td>
</tr>
<tr>
<td>options.recordId</td>
<td>string</td>
<td>optional</td>
<td>The record instance ID.</td>
</tr>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The ID of the action. For a list of action IDs, see the help topic <a href="#">Supported Record Actions</a>.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
</tbody>
</table>
### Error Code

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type.</td>
</tr>
<tr>
<td></td>
<td>– or –</td>
</tr>
<tr>
<td></td>
<td>The action exists, but cannot be executed on the specified record instance.</td>
</tr>
<tr>
<td>RECORD_DOES_NOT_EXIST</td>
<td>The specified record instance does not exist.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/action Module Script Samples](#).

```javascript
// Add additional code
...
var action = actionMod.get({recordType: 'timebill', id: 'approve'});
...
// Add additional code
```

**action.get.promise(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns an executable record action for the specified record type asynchronously. If the recordId parameter is specified, the action object is returned only if the specified action can be executed on the specified record instance.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Promise</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td></td>
<td>For additional information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td>Module</td>
<td>N/action Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>N/action Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.recordType</td>
<td>string</td>
<td>required</td>
<td>The record type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a list of record types, see record.Type.</td>
</tr>
<tr>
<td>options.recordId</td>
<td>string</td>
<td>optional</td>
<td>The record instance ID.</td>
</tr>
</tbody>
</table>
### N/action Module

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The ID of the action. For a list of action IDs, see the help topic Supported Record Actions.</td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS INVALID_RECORD_TYPE</td>
<td>The specified record type is invalid.</td>
</tr>
<tr>
<td>SSS MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>SSS INVALID_ACTION_ID</td>
<td>The specified action does not exist on the specified record type.</td>
</tr>
<tr>
<td></td>
<td>- or - The action exists, but cannot be executed on the specified record instance.</td>
</tr>
<tr>
<td>RECORD DOES NOT_EXIST</td>
<td>The specified record instance does not exist.</td>
</tr>
</tbody>
</table>

#### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/action Module Script Samples.

```javascript
// Add additional code
...
actionMod.get.promise({recordType: 'timebill', id: 'approve'}).then(function(action) {
    // do something with the action object
});
...
// Add additional code
```

### N/auth Module

Load the N/auth module when you want to change your NetSuite login credentials.

- **N/auth Module Members**
- **N/auth Module Script Sample**

#### N/auth Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>auth.###changeEmail###(options)</td>
<td>Server-side scripts</td>
<td>Changes the current user's NetSuite email address (user name).</td>
<td></td>
</tr>
</tbody>
</table>
Member Type | Name | Return Type / Value Type | Supported Script Types | Description
--- | --- | --- | --- | ---
auth.\#\#\#changePassword\#\#\#(options) | void | Server-side scripts | Changes the current user's NetSuite password.

### N/auth Module Script Sample

The following example changes the currently logged-in user's NetSuite email address and password.

This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record).

**Warning:** When you run this sample code in the SuiteScript Debugger, it logs a request to change the email and then changes the password.

For help with scripting in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/auth'],
function(auth) {
  function changeEmailAndPassword() {
    var password = 'myCurrentPassword';
    auth.changeEmail({
      password: password,
      newEmail: 'auth_test@newemail.com'
    });
    auth.changePassword({
      currentPassword: password,
      newPassword: 'myNewPa55Word'
    });
  }
  changeEmailAndPassword();
});
```

### auth.changeEmail(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to change the current user's NetSuite email address (user name).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts   For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>10 usage units</td>
</tr>
<tr>
<td>Module</td>
<td>N/auth Module</td>
</tr>
<tr>
<td>Since</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.password</td>
<td>string</td>
<td>required</td>
<td>The logged in user's NetSuite password.</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>options.newEmail</td>
<td>string</td>
<td>required</td>
<td>The logged in user's NetSuite email address.</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>options.onlyThisAccount</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, the email address change is applied only to roles within the current account. If set to false, the email address change is applied to all accounts and roles. The default value is true.</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_PSWD</td>
<td>The argument for options.password is invalid.</td>
<td></td>
</tr>
<tr>
<td>INVALID_EMAIL</td>
<td>The argument for options.newEmail is invalid.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/auth Module Script Sample.

```javascript
//Add additional code
...
auth.changeEmail({
  password: 'mypwd',
  newEmail: 'jwolf@netsuite.com',
  onlyThisAccount: true
});
...
//Add additional code
```

auth.changePassword(options)

Method Description
Method used to change the current user's NetSuite password.

Returns
void

Supported Script Types
Server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.currentPassword</td>
<td>string</td>
<td>required</td>
<td>The logged in user's current NetSuite password.</td>
<td>Version 2015 Release 2</td>
</tr>
<tr>
<td>options.newPassword</td>
<td>string</td>
<td>required</td>
<td>The logged in user's new NetSuite password.</td>
<td>Version 2015 Release 2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_PSWD</td>
<td>The argument for options.currentPassword is invalid.</td>
<td></td>
</tr>
<tr>
<td>USER_ERROR</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/auth Module Script Sample.

```javascript
//Add additional code
...
auth.changePassword({
    currentPassword: 'mycurrentPWD',
    newPassword: 'mynewPWD'
});
...
//Add additional code
```

N/cache Module

Load the cache module to enable temporary, short-term storage of data. Data is stored in the cache according to its specified time to live, or ttl. The ttl is specified in the Cache.put(options) method options.ttl parameter. The cache module is supported by all server-side script types.

Using a cache improves performance by eliminating the need for scripts in your account to retrieve the same piece of data more than one time. You can create a cache that is accessible at any of three levels: A cache can be available to the current script only, to all server-side scripts in the current bundle, or to all server-side scripts in your NetSuite account.

- N/cache Module Members
- Cache Object Members
## N/cache Module Script Sample

### N/cache Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>cache.Cache</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a segment of memory that can be used to temporarily store data on a short-term basis.</td>
</tr>
<tr>
<td>Method</td>
<td>cache.get(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Checks for a cache object with the specified name. If the cache exists, this method returns the cache object. If the cache does not exist, the system creates it.</td>
</tr>
<tr>
<td>Enum</td>
<td>cache.Scope</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>An enum used to populate the Cache.scope property.</td>
</tr>
</tbody>
</table>

### Cache Object Members

The following members are called on cache.Cache.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Cache.get(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Retrieves a value from the cache based on a key that you provide. If the requested value is not present or no longer in the cache, the method calls the user-defined function identified by the method's options.loader parameter. If the value provided by that function is not a string, the system uses JSON.stringify() to convert it. The string value is then cached and returned.</td>
</tr>
<tr>
<td></td>
<td>Cache.put(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Puts a value into the cache. If the value provided is not a string, the system uses JSON.stringify() to convert the value to a string. This data is not persistent.</td>
</tr>
<tr>
<td></td>
<td>Cache.remove(strings)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Removes a value from the cache.</td>
</tr>
<tr>
<td>Property</td>
<td>Cache.name</td>
<td>string</td>
<td>Server-side scripts</td>
<td>A label that identifies the cache.</td>
</tr>
<tr>
<td></td>
<td>Cache.scope</td>
<td>string</td>
<td>Server-side scripts</td>
<td>A value that describes the availability of the cache. A cache can be made available to the current script only, to all scripts in the current bundle, or to all scripts in your NetSuite account.</td>
</tr>
</tbody>
</table>
N/cache Module Script Sample

The following sample Suitelet retrieves the name of a city based on a ZIP code. To speed processing, the Suitelet uses a cache.

In this sample, ZIP code is the key used to retrieve city names from the cache. For any ZIP code provided, if the corresponding city value is not already stored in the cache, a loader function is called. This function, called zipCodeDatabaseLoader, loads a CSV file and uses it to find the requested value. (The zipCodeDatabaseLoader is shown in the next script sample.)

Note: This sample depends on a CSV file that must exist before the script is run. The sample CSV file is available here.

For help with scripting in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

Note: This sample script uses the define function. Note that you cannot use On Demand Debugging to step through a define function. You must use Deployed Debugging to step through this script.

```javascript
/**
 * @NScriptType Suitelet
 * @NApiVersion 2.x
 */
define(['N/cache', '/SuiteScripts/zipCodes/ca/zipToCityIndexCacheLoader'],
  function (cache, lib){
    const ZIP_CODES_CACHE_NAME = 'ZIP_CODES_CACHE';
    const ZIP_TO_CITY_IDX_JSON = 'ZIP_TO_CITY_IDX_JSON';

    function getZipCodeToCityLookupObj(){
      var zipCache = cache.getCache({name: ZIP_CODES_CACHE_NAME});
      var zipCacheJson = zipCache.get({
        key: ZIP_TO_CITY_IDX_JSON,
        loader: lib.zipCodeDatabaseLoader
      });
      return JSON.parse(zipCacheJson);
    }

    function findCityByZipCode(options){
      return getZipCodeToCityLookupObj()[String(options.zip)];
    }

    function onRequest(context){
      var start = new Date();
      if (context.request.parameters.purgeZipCache === 'true'){
        var zipCache = cache.getCache({name: ZIP_CODES_CACHE_NAME});
        zipCache.remove({key: ZIP_TO_CITY_IDX_JSON});
      }
      var cityName = findCityByZipCode({zip: context.request.parameters.zipcode});
      context.response.writeLine(cityName || 'Unknown :(');
      if (context.request.parameters.auditPerf === 'true'){
        context.response.writeLine('Time Elapsed: ' + (new Date().getTime() - start.getTime()) + ' ms');
      }
  })
```
N/cache Module

82

}
return {
onRequest: onRequest
};
});

The following custom module returns the loader function used in the preceding code sample. The
loader function shows how to use a CSV file to retrieve a value that was missing from a cache. This
script does not need to include logic for placing the retrieved value into the cache — whenever a value
is returned through the options.loader parameter, the value is automatically placed into the cache. For
this reason, a loader function can serve as the sole method of populating a cache with values.
/**
* @NApiVersion 2.0
* @NModuleScope Public
*/
define(['N/file', 'N/cache'], function(file, cache){
const ZIP_CODES_CSV_PATH = 'Resources/free-zipcode-ca-database-primary.csv';
function trimOuterQuotes(str){
return (str || '').replace(/^"+/, '').replace(/"+$/, '');
}
function zipCodeDatabaseLoader(context){
log.audit('Loading Zip Codes for ZIP_CODES_CACHE');
var zipCodesCsvText = file.load({id: ZIP_CODES_CSV_PATH}).getContents();
var zipToCityIndex = {};
var csvLines = zipCodesCsvText.split('\n');
util.each(csvLines.slice(1), function (el){
var cells = el.split(',');
var key = trimOuterQuotes(cells[0]);
var value = trimOuterQuotes(cells[2]);
if (parseInt(key, 10))
zipToCityIndex[String(key)] = value;
});
return zipToCityIndex;
}
return {
zipCodeDatabaseLoader : zipCodeDatabaseLoader
}
});

cache.Cache
Object Description

A segment of memory that can be used to temporarily store data needed by one
script, by all scripts in a bundle, or by all scripts in the NetSuite account.
This object is returned by cache.getCache(options).

Supported Script Types

Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Module

N/cache Module

SuiteScript 2.0 API Reference


## Cache Object Members

### Since

2016.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```javascript
// Add additional code
...
var myCache = cache.getCache({
  name: 'temporaryCache',
  scope: cache.Scope.PRIVATE
});
...
// Add additional code
```

**Cache.get(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>This method retrieves a string value from the cache. The value retrieved is identified by a key that you pass by using the options.key parameter. If a requested value is not present in the cache, the system calls the function identified by the options.loader parameter. This user-defined function should provide some logic for retrieving a value that is not in the cache. For an example, see N/cache Module Script Sample.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>String or null</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>1 unit if the value is present in the cache; 2 units if the loader function is used</td>
</tr>
<tr>
<td>Module</td>
<td>N/cache Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>cache.Cache</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Cache Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.key</td>
<td>string</td>
<td>required</td>
<td>A string that identifies the value to be retrieved from the cache. This value cannot be null.</td>
</tr>
<tr>
<td>options.loader</td>
<td>function</td>
<td>optional, but strongly recommended</td>
<td>A user-defined function that returns the requested value if it is not already present in the cache. Additionally, when the loader retrieves a value, the system automatically...</td>
</tr>
</tbody>
</table>
### N/cache Module

#### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>places that value in the cache. For this reason, NetSuite recommends using the loader function as the primary means of populating the cache. For an example, see N/cache Module Script Sample.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note also that if the value returned by the loader is not a string, the system uses JSON.stringify() to convert the value before it is placed in the cache and returned. The maximum size of a value that can be placed in the cache is 500KB. When no loader is specified and a value is missing from the cache, the system returns null.</td>
</tr>
<tr>
<td>options.ttl</td>
<td>number</td>
<td>optional</td>
<td>The maximum duration, in seconds, that a value retrieved by the loader can remain in the cache. Note that the value may be removed before the ttl limit is reached. The default ttl value is no limit. The minimum value is 300 (five minutes).</td>
</tr>
</tbody>
</table>

---

### Syntax

**Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```javascript
//Add additional code
...
var myValue = myCache.get({
  key: 'keyText',
  loader: loader,
  ttl: 18000
});
...
//Add additional code
```

### Cache.put(options)

**Method Description**: Use this method to place a value into a cache.

**Note**: You can also place a value in a cache by using the `Cache.get(options)` method and the options.loader parameter. In general, using the get method is recommended and may result in a more efficient design. For an example, see N/cache Module Script Sample.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Void</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All server-side scripts</td>
</tr>
</tbody>
</table>
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
1 unit

**Module**
N/cache Module

**Parent Object**
cache.Cache

**Sibling Object**
Cache Object Members

**Since**
2016.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.key</td>
<td>string</td>
<td>required</td>
<td>An identifier of the value that is being cached. The maximum size of the cache key is 4 kilobytes.</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>required</td>
<td>The value to place in the cache. If the value submitted is not a string, the system uses JSON.stringify() to convert the value before it is placed in the cache. The maximum size of the value is 500KB.</td>
</tr>
<tr>
<td>options.ttl</td>
<td>number</td>
<td>optional</td>
<td>The maximum duration, in seconds, that the value may remain in the cache. Note that the value may be removed before the ttl limit is reached. The default ttl value is no limit. The minimum value is 300 (five minutes).</td>
</tr>
</tbody>
</table>

**Important:** A cached value is not guaranteed to stay in the cache for the full duration of the ttl value. The ttl value represents the maximum time that the cached value may be stored. Cached data is not persistent, and it is recommended that you use the Cache.get(options) method and options.loader parameter to set and retrieve data.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```javascript
//Add additional code
...
myCache.put({
    key: 'keyText',
    value: 'valueText',
    ttl: 300
});
...```
Cache.remove(options)

Method Description: Removes a value from the cache.

Returns: Void

Supported Script Types: All server-side scripts

Governance: 1 unit

Module: N/cache Module

Parent Object: cache.Cache

Sibling Object Members: Cache Object Members

Since: 2016.2

Parameters:

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.key</td>
<td>string</td>
<td>required</td>
<td>An identifier of the value that is being removed.</td>
</tr>
</tbody>
</table>

Syntax:

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```
//Add additional code
...
myCache.remove({
    key: 'keyText'
});
...
//Add additional code
```

Cache.name

Property Description: A label that identifies a cache.

Type: string

Supported Script Types: All server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.
Module | N/cache Module
---|---
Parent Object | cache.Cache
Sibling Object Members | Cache Object Members
Since | 2016.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```javascript
//Add additional code
...
var myCache = cache.getCache({
    name: 'temporaryCache',
    scope: cache.Scope.PRIVATE
});
...
//Add additional code
```

Cache.scope

<table>
<thead>
<tr>
<th>Property Description</th>
<th>A label that describes the availability of the cache to other scripts.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>cache.Scope</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All server-side scripts</td>
</tr>
<tr>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
</tbody>
</table>

Module | N/cache Module
---|---
Parent Object | cache.Cache
Sibling Object Members | Cache Object Members
Since | 2016.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```javascript
//Add additional code
...
var myCache = cache.getCache({
    name: 'temporaryCache',
    scope: cache.Scope.PRIVATE
});
...
//Add additional code
```
cache.getCache(options)

**Method Description**
Method used to create a new cache.Cache object.

**Returns**
cache.Cache

**Supported Script Types**
All server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
n/a

**Module**
N/cache Module

**Sibling Module Members**
N/cache Module Members

**Since**
2016.2

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>required</td>
<td>A label that will identify the cache you are creating. The maximum size of the cache name is 1 kilobyte.</td>
</tr>
<tr>
<td>options.scope</td>
<td>string</td>
<td>optional, but if you do not set a value, the default of PRIVATE is used</td>
<td>This value is set with the cache.Scope enum. It determines the availability of the cache. A cache can be made available to the current script only, to all scripts in the current bundle, or to all scripts in your NetSuite account.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```javascript
//Add additional code
...
var myCache = cache.getCache({
    name: 'temporaryCache',
    scope: cache.Scope.PRIVATE
});
...
//Add additional code
```

**cache.Scope**

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration that holds string values that describe the availability of the cache. This enum is used to set the value of the Cache.scope property.</td>
<td></td>
</tr>
</tbody>
</table>
N/cache Module

JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>Enum</th>
</tr>
</thead>
</table>

Supported Script Types

All server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Module

N/cache Module

Sibling Module Members

N/cache Module Members

Since

2016.2

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRIVATE</td>
<td>The cache is available only to the current script. This value is the default.</td>
</tr>
<tr>
<td>PROTECTED</td>
<td>The cache is available only to some scripts, as follows:</td>
</tr>
<tr>
<td></td>
<td>■ If the script is part of a bundle, the cache is available to all scripts in the same bundle.</td>
</tr>
<tr>
<td></td>
<td>■ If the script is not in a bundle, then the cache is available to all scripts not in a bundle.</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>The cache is available to any script in the NetSuite account.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/cache Module Script Sample.

```javascript
//Add additional code
...
var myCache = cache.getCache({
    name: 'temporaryCache',
    scope: cache.Scope.PRIVATE
});
...  //Add additional code
```

N/certificateControl Module

The N/certificateControl module enables scripting access to the Digital Certificates list found in the UI at Setup > Company > Certificates. You can use this module to find the correct certificate for a subsidiary and check the file type. For more information, see the help topics Digital Signing and Uploading Digital Certificates.
In order to access this module, you must use the Execute As Role field on the script deployment record. Select either the Administrator role or a custom role with the Certificate Access permission. For more information, see the help topic Access to Digital Certificates.

N/certificateControl Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>certificateControl.findCertificates(options)</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Returns metadata about the certificate(s).</td>
</tr>
<tr>
<td>Enum</td>
<td>certificateControl.Type</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Enum for certificate types. PFX and P12 are supported types.</td>
</tr>
</tbody>
</table>

N/certificateControl Module Script Samples

Example 1

The following example shows how to filter the Digital Certificates list by subsidiary and by file type.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/certificateControl'],
function(certificateControl){
    var all = certificateControl.findCertificates();
    var specificType = certificateControl.findCertificates({
        type: 'PFX'
    });
    var specificSub = certificateControl.findCertificates({
        subsidiary: 93
    });
    var specificTypeAndSub = certificateControl.findCertificates({
        type: 'PFX',
        subsidiary: 93
    });
});
```

certificateControl.findCertificates(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Metadata about the certificate(s)</td>
<td>Server-side scripts</td>
<td>10 units</td>
<td>N/certificateControl Module</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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>options.subsidiary</td>
<td>number</td>
<td>optional</td>
<td>The internal ID of the subsidiary.</td>
<td>2019.1</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>optional</td>
<td>The certificate file type.</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.

Related Topics

- Digital Signing
- Uploading Digital Certificates
- N/https/clientCertificate Module
- N/crypto/certificate Module
- SuiteScript 2.0 Modules
- SuiteScript 2.0

**certificateControl.Type**

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enum for the certificate file type. PFX and P12 are supported.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/certificateControl Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All 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>2019.1</td>
</tr>
</tbody>
</table>

**Values**

- PFX
- P12

Related Topics

- Digital Signing
- N/https/clientCertificate Module
- N/crypto/certificate Module
- SuiteScript 2.0 Modules
- SuiteScript 2.0
N/config Module

Load the N/config module when you want to access NetSuite configuration settings. The `config.load(options)` method returns a `record.Record` object. Use the `record.Record` object members to access configuration settings. You do not need to load the record module to do this.

See `config.Type` for a list of supported configuration objects.

- N/config Module Members
- N/config Module Script Sample

N/config Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>config.load(options)</td>
<td>record.Record</td>
<td>Server-side scripts</td>
<td>Loads a <code>record.Record</code> object that encapsulates the specified configuration page.</td>
</tr>
<tr>
<td>Enum</td>
<td>config.Type</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported configuration objects. This enum is used to set the value of the NetSuite configuration page you want to access.</td>
</tr>
</tbody>
</table>

N/config Module Script Sample

Note: This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

This example loads the Company Information configuration page. It then sets the values specified for the Tax ID Number field and the Employer Identification Number field.

Note: The IDs in this sample are placeholders. Replace the Tax ID Number field and the Employer Identification Number with valid IDs from your NetSuite account.

For help with scripting in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/config'],
    function(config) {
        function setTaxAndEmployerId() {
            var companyInfo = config.load({
                type: config.Type.COMPANY_INFORMATION
            });
            companyInfo.setValue({
                fieldId: 'taxid',
                value: '1122334455'
            });
        }
    });
```
```javascript
companyInfo.setValue({
  fieldId: 'employerid',
  value: '123456789'
});
companyInfo.save();
companyInfo = config.load({
  type: config.Type.COMPANY_INFORMATION
});
var taxid = companyInfo.getValue({
  fieldId: 'taxid'
});
setTaxAndEmployerId();
```

### config.load(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to load a <code>record.Record</code> object that encapsulates the specified NetSuite configuration page. After the configuration page loads, all preference names and IDs are available to get or set. For more information, see the help topic Preference Names and IDs. You can use the following <code>Record</code> Object Members to get and set preference names and IDs:</th>
</tr>
</thead>
</table>
| | ▪ `Record.getField(options)`  
| | ▪ `Record.getFields()`  
| | ▪ `Record.getText(options)`  
| | ▪ `Record.getValue(options)`  
| | ▪ `Record.setText(options)`  
| | ▪ `Record.setValue(options)`  |
| Returns | `record.Record` |
| Supported Script Types | Server-side scripts  
For additional information, see the help topic SuiteScript 2.0 Script Types. |
| Governance | 10 usage units |
| Module | `N/config Module` |
| Since | 2015.2 |

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>enum</td>
<td>required</td>
<td>The NetSuite configuration page you want to access. Use the <code>config.Type</code> enum to set the value.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### 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>options.isDynamic</td>
<td>boolean</td>
<td>true / false</td>
<td>Determines whether the record is loaded in dynamic mode.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to true, the record is loaded in dynamic mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to false, the record is loaded in standard mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 – Standard and Dynamic Modes.</td>
<td></td>
</tr>
</tbody>
</table>

### Error Code

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_RCRD_TYPE</td>
<td>The record type (type) is invalid.</td>
<td>The type argument is invalid or missing.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/config Module Script Sample.

```javascript
//Add additional code
...
var configRecObj = config.load({
   type: config.Type.COMPANY_INFORMATION
});
configRecObj.setText({
    fieldId: 'fiscalmonth',
    text: 'July'
});
configRecObj.save();
...
//Add additional code
```

### config.Type

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for supported configuration pages.</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Configuration Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>USER_PREFERENCES</td>
<td>Set Preferences page (Home &gt; Set Preferences)</td>
</tr>
<tr>
<td></td>
<td>For more information about the fields on the page, see the help topic User Preferences.</td>
</tr>
<tr>
<td>COMPANY_INFORMATION</td>
<td>Company Information page (Setup &gt; Company &gt; Company Information)</td>
</tr>
<tr>
<td></td>
<td>For more information about the fields on the page, see the help topic Company Information.</td>
</tr>
<tr>
<td>COMPANY_PREFERENCES</td>
<td>General Preferences page (Setup &gt; Company &gt; General Preferences)</td>
</tr>
<tr>
<td></td>
<td>For more information about the fields on the page, see the help topic General Preferences.</td>
</tr>
<tr>
<td>ACCOUNTING_PREFERENCES</td>
<td>Accounting Preferences page (Setup &gt; Accounting &gt; Accounting Preferences)</td>
</tr>
<tr>
<td></td>
<td>For more information about the fields on the page, see the help topic Accounting Preferences.</td>
</tr>
<tr>
<td>ACCOUNTING_PERIODS</td>
<td>Accounting Periods page (Setup &gt; Accounting &gt; Manage Accounting Periods)</td>
</tr>
<tr>
<td></td>
<td>For more information about the fields on the page, see the help topic Accounting Periods.</td>
</tr>
<tr>
<td>TAX_PERIODS</td>
<td>Tax Periods page (Setup &gt; Accounting &gt; Manage Tax Periods)</td>
</tr>
<tr>
<td></td>
<td>For more information about the fields on the page, see the help topic Tax Periods.</td>
</tr>
<tr>
<td>FEATURES</td>
<td>Enable Features page (Setup &gt; Company &gt; Enable Features)</td>
</tr>
<tr>
<td></td>
<td>For more information about feature names and IDs, see the help topic Feature Names and IDs.</td>
</tr>
<tr>
<td>TIME_POST</td>
<td>For additional information, see the help topic Posting Time Transactions.</td>
</tr>
<tr>
<td>TIME_VOID</td>
<td>For additional information, see the help topic Posting Time Transactions.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/config Module Script Sample.

```javascript
//Add additional code
...
var configRecObj = config.load({
    type: config.Type.COMPANY_INFORMATION
});
configRecObj.setText({
    fieldId: 'fiscalmonth',
    text: 'July'
});
configRecObj.save();
...
//Add additional code
```
The N/crypto module encapsulates hashing, hash-based message authentication (hmac), and symmetrical encryption.

When the crypto module is used, SuiteScript also loads N/encode Module.

- **N/crypto Module Members**
  - Cipher Object Members
  - CipherPayload Object Members
  - Decipher Object Members
  - Hash Object Members
  - Hmac Object Members
  - SecretKey Object Members
  - N/crypto Module Script Samples

### N/crypto Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>crypto.Cipher</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a cipher.</td>
</tr>
<tr>
<td></td>
<td>crypto.CipherPayload</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a cipher payload.</td>
</tr>
<tr>
<td></td>
<td>crypto.Decipher</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a decipher.</td>
</tr>
<tr>
<td></td>
<td>crypto.Hash</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a hash.</td>
</tr>
<tr>
<td></td>
<td>crypto.SecretKey</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a secret key handle.</td>
</tr>
<tr>
<td>Method</td>
<td>crypto.createCipher</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Creates and returns a new crypto.Cipher Object.</td>
</tr>
<tr>
<td>Enum</td>
<td>crypto.EncryptionAlg</td>
<td>string (readonly)</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported encryption algorithms. Sets the options.algorithm parameter for crypto.createCipher(options).</td>
</tr>
<tr>
<td></td>
<td>crypto.HashAlg</td>
<td>string (readonly)</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported hashing algorithms. Sets the options.algorithm parameter for crypto.createHash(options) and crypto.createHmac(options).</td>
</tr>
<tr>
<td></td>
<td>crypto.Padding</td>
<td>string (readonly)</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported cipher padding. Sets the</td>
</tr>
</tbody>
</table>

**N/crypto Module Script Samples**

**Scenario:**

- **Encrypting Data**
  ```javascript
  var cipher = crypto.createCipher('aes128-cbc', 'my-secret-key');
  var encryptedData = cipher.update('Hello, World!', 'utf-8');
  encryptedData += cipher.final();
  ```

- **Validating Data with HMAC**
  ```javascript
  var hmac = crypto.createHmac('sha256', 'my-secret-key');
  var data = 'Hello, World!';
  var hmacValue = hmac.update(data).digest('hex');
  ```
### Crypto Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>options.padding</td>
<td></td>
<td></td>
<td>parameter for crypto.createCipher(options) and crypto.createDecipher(options).</td>
</tr>
</tbody>
</table>

#### Cipher Object Members

The following members are called on `crypto.Cipher`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Cipher.update(options)</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Updates the clear data with the specified encoding</td>
</tr>
<tr>
<td>Method</td>
<td>Cipher.final(options)</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Returns the cipher data.</td>
</tr>
</tbody>
</table>

#### CipherPayload Object Members

The following members are called on `crypto.CipherPayload`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>CipherPayload.ciphertext</td>
<td>string</td>
<td>Server-side scripts</td>
<td>The result of the ciphering process.</td>
</tr>
<tr>
<td>Property</td>
<td>CipherPayload.iv</td>
<td>number</td>
<td>Server-side scripts</td>
<td>An initialization vector</td>
</tr>
</tbody>
</table>

#### Decipher Object Members

The following members are called on `crypto.Decipher`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Decipher.final(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns the clear data.</td>
</tr>
<tr>
<td>Method</td>
<td>Decipher.##update(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Updates cipher data with the specified encoding.</td>
</tr>
</tbody>
</table>

#### Hash Object Members

The following members are called on `crypto.Hash`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Hash.digest(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Calculates the digest of the data to be hashed.</td>
</tr>
<tr>
<td>Method</td>
<td>Hash.update(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Updates the clear data with the encoding specified.</td>
</tr>
</tbody>
</table>
Hmac Object Members

The following members are called on crypto.Hmac.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Hmac.digest(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Gets the computed digest.</td>
</tr>
<tr>
<td></td>
<td>Hmac.update(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Updates the clear data with the encoding specified.</td>
</tr>
</tbody>
</table>

SecretKey Object Members

The following members are called on crypto.SecretKey.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>SecretKey.guid</td>
<td>string</td>
<td>Server-side scripts</td>
<td>The GUID associated with the secret key.</td>
</tr>
<tr>
<td></td>
<td>SecretKey.encoding</td>
<td>string</td>
<td>Server-side scripts</td>
<td>The encoding used for the clear text value of the secret key.</td>
</tr>
</tbody>
</table>

N/crypto Module Script Samples

Example 1

The following example demonstrates the APIs needed to generate a secure key using the SHA512 hashing algorithm. It is not a functional example that will work in the debugger (because the GUID does not exist in your account). Refer to the Suitelet example for a more complete usage. See Example 2.

To create a real password GUID, obtain a password value from a credential field on a form. For more information, see Form.addCredentialField(options). Also see N/https Module Script Sample for a Suitelet example that shows creating a form field that generates a GUID.

**Note:** The GUID in this sample is a placeholder. You must replace it with a valid value from your NetSuite account.

For help with scripting in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

```javascript
/**
 * @NApiVersion 2.x
 */
require(["N/crypto", "N/encode", "N/runtime"],
function(crypto, encode, runtime) {
    function createSecureKeyWithHash() {
        function createSecureKeyWithHash() {
            var inputString = "TWJ3ED0n5w=";
            var myGuid = '{284CF6D25B1D76FE9D150207E49DF}';
            var sKey = crypto.createSecretKey({
                guid: myGuid,
                encoding: encode.Encoding.UTF_8
            });
```
var hmacSHA512 = crypto.createHmac({
    algorithm: crypto.HashAlg.SHA512,
    key: sKey
});
hmacSHA512.update({
    input: inputString,
    inputEncoding: encode.Encoding.BASE_64
});
var digestSHA512 = hmacSHA512.digest({
    outputEncoding: encode.Encoding.HEX
});
}
createSecureKeyWithHash();

Example 2

Note: This sample script uses the define function. Note that you cannot use On Demand Debugging to step through a define function. You must use Deployed Debugging to step through this script.

Important: The default maximum length for a secret key field is 32 characters. If needed, use the Field.maxLength property to change this value.

/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define(['N/ui/serverWidget', 'N/runtime', 'N/crypto', 'N/encode'],
    function(ui, runtime, crypto, encode) {
        function onRequest(option) {
            if (option.request.method === 'GET') {
                var form = ui.createForm({
                    title: 'My Credential Form'
                });
                var skField = form.addSecretKeyField({
                    id: 'mycredential',
                    label: 'Credential',
                    restrictToScriptIds: [runtime.getCurrentScript().id],
                    restrictToCurrentUser: false
                })
                skField.maxLength = 200;
                form.addSubmitButton();
                option.response.writePage(form);
            } else {
                var form = ui.createForm({
                    title: 'My Credential Form'
                });
                var inputString = "YWJjZGVmZwo=";
                var myGuid = option.request.parameters.mycredential;
                // Create the key
                var sKey = crypto.createSecretKey({
                    guid: myGuid,
                    encoding: encode.Encoding.UTF_8
                })
            }
        }
    })
)
crypto.Cipher

Object Description
Encapsulates a cipher.

For a complete list of this object’s methods and properties, see Cipher Object Members.

Supported Script Types
Server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

Module
N/crypto Module

Since
2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...```
var cipher = crypto.createCipher({
    algorithm: crypto.EncryptionAlg.AES,
    key: sKey
  });
...
//Add additional code

Cipher.final(options)

<table>
<thead>
<tr>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method used to return the cipher data.</td>
</tr>
<tr>
<td>Sets the output encoding for the crypto.CipherPayload object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>A crypto.CipherPayload Object</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server-side scripts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/crypto Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.outputEncoding</td>
<td>enum</td>
<td>optional</td>
<td>The output encoding for a crypto.CipherPayload object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default value is HEX.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the encode.Encoding enum to set the value.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
crypto.createCipher({
    algorithm: crypto.EncryptionAlg.AES,
    key: sKey
  });
...
var cipherPayload = cipher.final({
    outputEncoding: encode.Encoding.BASE_64
  });
...
//Add additional code
```
Cipher.update(options)

Method Description | Method used to update the clear data with the specified encoding.
--- | ---
Returns | Void
Supported Script Types | Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.
Governance | None
Module | N/crypto Module
Since | 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.input</td>
<td>string</td>
<td>required</td>
<td>The clear data to be updated.</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>enum</td>
<td>optional</td>
<td>The input encoding. Use the encode.Encoding enum to set the value. The default value is UTF_8.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var reencoded = Cipher.update({
    input: 'Carrot cake gummi bears'
});
...
//Add additional code
```

crypto.CipherPayload

Object Description | Encapsulates a cipher payload.
For a complete list of this object's methods and properties, see CipherPayload Object Members.

Supported Script Types | Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Module | N/crypto Module
Since 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
crypto.createCipher({
    algorithm: crypto.EncryptionAlg.AES,
    key: sKey
});

var cipherPayload = cipher.final({
    outputEncoding: encode.Encoding.HEX
});
...
//Add additional code
```

**CipherPayload.ciphertext**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The result of the ciphering process. For example, to take the cipher payload and send it to another system.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
</table>

**Supported Script Types**

- Server-side scripts
  - For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/crypto Module

**Since**

2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
log.debug({
    title: "Ciphertext: ",
    details: cipherPayload.ciphertext
});
...
//Add additional code
```

**CipherPayload.iv**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initialization vector for the cipher payload.</td>
</tr>
</tbody>
</table>
You can pass in the iv value to `crypto.createDecipher(options)`

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/crypto Module Script Samples](#).

```javascript
//Add additional code
...
log.debug({
  title: "CipherPayload IV: ",
  details: cipherPayload.iv
});
...
//Add additional code
```

crypto.Decipher

**Object Description**

Encapsulates a decipher. This object has methods that decrypt.

For a complete list of this object's methods and properties, see [Decipher Object Members](#).

**Supported Script Types**

Server-side scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

<table>
<thead>
<tr>
<th>Module</th>
<th>N/crypto Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/crypto Module Script Samples](#).

```javascript
//Add additional code
...
crypto.createDecipher({
  algorithm: crypto.EncryptionAlg.AES,
  key: sKey
});
...
//Add additional code
```
Decipher.final(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to return the clear data.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.outputEncoding</td>
<td>string</td>
<td>optional</td>
<td>Specifies the encoding for the output</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set the value using the encode.Encoding enum.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default value is UTF_8.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var decipher1 = Decipher.final({
    outputEncoding: encode.Encoding.HEX
});
...
//Add additional code
```

Decipher.update(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to update cipher data with the specified encoding.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.input</td>
<td>string</td>
<td>required</td>
<td>The data to update</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>string</td>
<td>optional</td>
<td>Specifies the encoding of the input data</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set the value using the <code>encode.Encoding</code> enum. The default value is HEX.</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.

Syntax

```javascript
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

//Add additional code
...
var decipher1 = Decipher.update({
    input: '73616d706c65737472696e67',
    inputEncoding: encode.Encoding.HEX
});
...
//Add additional code
```

crypto.Hash

**Object Description**

Encapsulates a hash.

For a complete list of this object's methods and properties, see Hash Object Members.

**Supported Script Types**

Server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/crypto Module

**Since**

2015.2

Syntax

```javascript
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

//Add additional code
...
var hashObj = crypto.createHash({
    algorithm: crypto.HashAlgorithm.SHA256
});
...
//Add additional code
```
Hash.digest(options)

Method Description
Calculates the digest of the data to be hashed.

Returns
A hash value as a string

Supported Script Types
Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/crypto Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.outputEncoding</td>
<td>string</td>
<td>optional</td>
<td>The output encoding. Set using the encode.Encoding enum. The default value is HEX.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var digestSample = hashObj.digest({
    outputEncoding: encode.Encoding.HEX
});
...
//Add additional code
```

Hash.update(options)

Method Description
Method used to update clear data with the encoding specified.

Returns
Void

Supported Script Types
Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/crypto Module

Since
2015.2
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.input</td>
<td>string</td>
<td>required</td>
<td>The data to be updated.</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>string</td>
<td>optional</td>
<td>The input encoding. Set using the encode.Encoding enum. The default value is UTF_8.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var inputString = 'Lemon drops ice cream jelly marzipan cake';
hashSample.update({
  input: inputString
});
...
//Add additional code
```

crypto.Hmac

Object Description
Encapsulates an hmac.

For a complete list of this object’s methods and properties, see Hmac Object Members.

Supported Script Types
Server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

Module
N/crypto Module

Since
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var hmacSHA512 = crypto.createHmac({
  algorithm: crypto.HashAlg.SHA512,
  key: sKey
});
...
//Add additional code
```
Hmac.digest(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the computed digest.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>An hmac value as a string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>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>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.outputEncoding</td>
<td>string</td>
<td>optional</td>
<td>Specifies the encoding of the output string. Set using the encode.Encoding enum. The default value is HEX.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var digestSHA512 = hmacSHA512.digest({
   outputEncoding: encode.Encoding.HEX
});
...
//Add additional code
```

Hmac.update(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to update the clear data with the encoding specified.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>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>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.input</td>
<td>string</td>
<td>required</td>
<td>The hmac data to be updated.</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>enum</td>
<td>optional</td>
<td>The input encoding. Set using the encode.Encoding enum. The default value is UTF_8.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var hmacSHA512 = crypto.createHMAC('sha512', secretKey);    
  hmacSHA512.update({
    input: inputString,  
    inputEncoding: encode.Encoding.BASE_64
  });  
...
//Add additional code
```

crypto.SecretKey

Object Description

Encapsulates the handle to the key. The handler does not store the key value. It points to the key stored within the NetSuite system. The GUID is also required to find the key.

For a complete list of this object's methods and properties, see SecretKey Object Members.

Supported Script Types

Server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

Module

N/crypto Module

Since

2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var sKey = crypto.createSecretKey({
  guid: '284CFB2D225B1D76FB94D150207E49DF',  
  encoding: encode.Encoding.UTF_8
});
...
//Add additional code
```
SecretKey.encoding

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The encoding used for the clear text value of the secret key.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>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>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```
//Add additional code
...
log.debug({
  title: 'Secret Key Encoding: ',
  details: sKey.encoding
});
...
//Add additional code
```

Secretkey.guid

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The GUID associated with the secret key.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>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>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```
//Add additional code
...
log.debug({
  title: 'Secret Key GUID: ',
  details: sKey.guid
});
...
//Add additional code
```
crypto.createCipher(options)

Method Description
Method used to create and return a crypto.EncryptionAlg object.

Note:
The blockCipherMode is automatically set to CBC.

Returns
A crypto.EncryptionAlg object

Supported Script Types
Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/crypto Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.algorithm</td>
<td>string</td>
<td>required</td>
<td>The hash algorithm. Set the value using the crypto.EncryptionAlg enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.key</td>
<td>object</td>
<td>required</td>
<td>The crypto.SecretKey object.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: When using the crypto.SecretKey object for an AES algorithm, the length of the text (secret key) that is used to generate the GUID must be 16, 24, or 32 characters.</td>
<td></td>
</tr>
<tr>
<td>options.padding</td>
<td>string</td>
<td>optional</td>
<td>The padding for the cipher text.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set the value using the crypto.Padding enum. By default, the value is set to PKCS5Padding.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
//Add additional code
var cipher = crypto.createCipher({
  algorithm: crypto.EncryptionAlg.AES,
```
crypto.createDecipher(options)

**Method Description**
Method used to create a crypto.Decipher object.

![Note:](image) The blockCipherMode is automatically set to CBC.

**Returns**
A crypto.Decipher object.

**Supported Script Types**
Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/crypto Module

**Since**
2015.2

**Parameters**

![Note:](image) The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.algorithm</td>
<td>string</td>
<td>required</td>
<td>The hash algorithm. Set by the crypto.EncryptionAlg enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.key</td>
<td>object</td>
<td>required</td>
<td>The crypto.SecretKey object used for encryption.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.padding</td>
<td>object</td>
<td>optional</td>
<td>The padding for the cipher. Set the value using the crypto.Padding enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.iv</td>
<td>string</td>
<td>required</td>
<td>The initialization vector that was used for encryption.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

![Important:](image) The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
```
var decipher = crypto.createDecipher({'algorithm': crypto.EncryptionAlg.AES, 'key': sKey, 'padding': NoPadding, 'iv': '2311141720'});
...
//Add additional code

crypto.createHash(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to create a crypto.Hash object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>The crypto.Hash object created using this method.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.algorithm</td>
<td>string</td>
<td>required</td>
<td>- The hash algorithm. Set using the crypto.HashAlg enum.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```
//Add additional code
...
var hashObj = crypto.createHash({
    algorithm: crypto.HashAlg.SHA256
});
...
//Add additional code
```

crypto.createHmac(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to create a crypto.Hmac object.</th>
</tr>
</thead>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.algorithm</td>
<td>string</td>
<td>required</td>
<td>The hash algorithm. Use the crypto.HashAlg enum to set this value.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.key</td>
<td>object</td>
<td>required</td>
<td>The crypto.SecretKey object.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var hmacObj = crypto.createHmac({
   algorithm: HashAlg.SHA256,
   key: sKey
});
...
//Add additional code
```

crypto.createSecretKey(options)

**Method Description**

Method used to create a new crypto.SecretKey object.

This method can take a GUID. Use Form.addCredentialField(options) to generate a value.

**Note:** When using the crypto.SecretKey object for an AES algorithm, the length of the text (secret key) that is used to generate the GUID must be 16, 24, or 32 characters.

<table>
<thead>
<tr>
<th>Returns</th>
<th>A crypto.SecretKey object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
</tbody>
</table>
Since 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.guid</td>
<td>string</td>
<td>required</td>
<td>A GUID used to generate a secret key. The GUID can resolve to either data or metadata.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.encoding</td>
<td>enum</td>
<td>optional</td>
<td>Specifies the encoding for the SecureKey. Set this value using the <code>encode.Encoding</code> enum. The default value is HEX.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
// Add additional code
...
var secretKey = crypto.createSecretKey({
  encoding: encode.Encoding.HEX,
  guid: '284CFB2D225B1D76FB94D150207E49DF'
});
...
// Add additional code
```

crypto.EncryptionAlg

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported encryption algorithms. Sets the <code>options.algorithm</code> parameter for <code>crypto.createCipher(options)</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

Module | N/crypto Module
---|---
Supported Script Types | All server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.
Since | 2015.2
Values

- AES

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var cipher = crypto.createCipher({
    algorithm: crypto.EncryptionAlg.AES,
    key: sKey
});
...
//Add additional code
```

crypto.HashAlg

| Enum Description | Holds the string values for supported hashing algorithms. Sets the value of the `options.algorithm` parameter for `crypto.createHash(options)` and `crypto.createHmac(options)`.
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/crypto Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All server-side scripts</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Values

- SHA1
- SHA256
- SHA512
- MD5

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/crypto Module Script Samples.

```javascript
//Add additional code
...
```
```javascript
var hmacSHA512 = crypto.createHmac(
    {    
        algorithm: crypto.HashAlg.SHA512,
        key: sKey
    });
...
//Add additional code
```

### crypto.Padding

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported cipher padding. Sets the <code>options.padding</code> parameter for <code>crypto.createCipher(options)</code> and <code>crypto.createDecipher(options)</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/crypto Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All server-side scripts</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Values

- NoPadding
- PKCSSPadding

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/crypto Module Script Samples`.

```javascript
//Add additional code
...
var cipher = crypto.createCipher({
    algorithm: crypto.EncryptionAlg.AES,
    key: sKey,
    padding: crypto.Padding.NoPadding
});
...
//Add additional code
```

### N/crypto/certificate Module

Load the `N/crypto/certificate` module to sign XML documents or strings with digital certificates using asymmetric cryptography. In addition to signing XML documents, you can create signer and verifier objects and verify signed documents with this module.
The N/crypto/certificate Module includes:

- **N/crypto/certificate Module Members**
- **Signer Object Members**
- **Verifier Object Members**
- **SignedXml Object Members**
- **N/crypto/certificate Module Script Samples**

### N/crypto/certificate Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>certificate.SignedXml</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a signed XML string that has been digitally signed. Use certificate.signXml(options) to create this object.</td>
</tr>
<tr>
<td></td>
<td>certificate.Signer</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Object for creating signatures for plain strings. Use certificate.createSigner(options) to create this object.</td>
</tr>
<tr>
<td></td>
<td>certificate.Verifier</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Object for verifying plain string signatures. Use certificate.createVerifier(options) to create this object.</td>
</tr>
</tbody>
</table>

#### Method

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificate.verifyXmlSignature</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Verifies the signature in the signedXml file.</td>
</tr>
<tr>
<td>certificate.signXml(options)</td>
<td>certificate.SignedXml</td>
<td>Server-side scripts</td>
<td>Signs inputXml string using certId. Returns SignedXml as string.</td>
</tr>
</tbody>
</table>

#### Enum

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>certificate.HashAlg</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Enum for hash algorithms. SHA1, SHA256, SHA384, or SHA512 are supported digest methods and values for this enum.</td>
</tr>
</tbody>
</table>

### Signer Object Members

The signer object creates signatures for plain strings. The following members are called on the certificate.Signer object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Signer.update(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Updates the string to be signed.</td>
</tr>
</tbody>
</table>
Verifier Object Members

The following members are called on the `certificate.Verifier` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Verifier.#update(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Updates string to be verified against specified certificate.</td>
</tr>
<tr>
<td>Method</td>
<td>Verifier.verify(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Verifies string against provided signature using specified certificate.</td>
</tr>
</tbody>
</table>

SignedXml Object Members

The following members are called on the `certificate.SignedXml` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>SignedXml.asString()</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns signed xml as a string.</td>
</tr>
</tbody>
</table>

N/crypto/certificate Module Script Samples

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

Example 1

The following example loads an XML file from the File Cabinet and signs it using the digital certificate with internal ID ‘custcerticate1’.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/crypto/certificate'],
```
function (cert, file) {
   var infNFe = file.load({
      id: 922
   });
   var signedXml = cert.signXML({
      algorithm: 'SHA1',
      certId: 'custcertificate1',
      rootTag: 'infNFe',
      xmlString: infNFe.getContents()
   });
   certificate.verifyXMLSignature({
      signedXml: signedXml,
      rootTag: 'infNFe'
   });
}

Example 2

The following example creates a certificate.Signer object, signs it, and then creates a certificate.Verifier object and verifies the object.

Note: This sample script uses the require function so that you can copy it into the debugger and test it. Keep in mind that you must use the define function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
require(['N/crypto/certificate'], function (certificate) {
   var signer = certificate.createSigner({
      certId: 'custcertificate1',
      algorithm: 'SHA1'
   });
   signer.update('test');
   var result = signer.sign();
   var verifier = certificate.createVerifier({
      certId: 'custcertificate1',
      algorithm: 'SHA1'
   });
   verifier.update('test');
   verifier.verify(result);
});
```

certificate.SignedXml

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates a signed XML string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
</tbody>
</table>

For additional information, see the help topic SuiteScript 2.0 Script Types.
<table>
<thead>
<tr>
<th>Module</th>
<th>N/crypto/certificate Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Methods and Properties</td>
<td>SignedXml Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**SignedXml.asString()**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to return the signed XML as a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto/certificate Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>certificate.SignedXml</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>SignedXml Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**certificate.Signer**

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Object used for signing plain strings. This object is returned by the certificate.createSigner(options) method.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto/certificate Module</td>
</tr>
<tr>
<td>Methods and Properties</td>
<td>Signer Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**Signer.update(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Updates the input string to be signed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/crypto/certificate Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>certificate.Signer</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.input</td>
<td>string</td>
<td>required</td>
<td>The string to update.</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>string</td>
<td>optional</td>
<td>Encoding of the string to sign. The default value is UTF-8.</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.

#### Signer.sign(options)

**Method Description**: Signs the string and return the signature.  
**Returns**: string  
**Supported Script Types**: Server-side scripts  
For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**: None  
**Module**: N/crypto/certificate Module  
**Parent Object**: certificate.Signer  
**Sibling Object Members**: Signer Object Members  
**Since**: 2019.1

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.outputEncoding</td>
<td>string</td>
<td>optional</td>
<td>Encoding of the signed string in Base64 format.</td>
</tr>
</tbody>
</table>

#### certificate.Verifier

**Object Description**: Encapsulates an object for verifying plain string signatures.  
**Supported Script Types**: Server-side scripts  
For additional information, see the help topic [SuiteScript 2.0 Script Types](#).
## N/crypto/certificate Module

### Methods and Properties

#### Verifier Object Members

**Since** 2019.1

### Verifier.update(options)

**Method Description**
Updates the string to be verified against a specified certificate.

**Returns**
void

**Supported Script Types**
Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.input</td>
<td>string</td>
<td>required</td>
<td>The string to verify.</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>string</td>
<td>optional</td>
<td>Encoding of the string to verify. The default value is UTF-8.</td>
</tr>
</tbody>
</table>

### Verifier.verify(options)

**Method Description**
Verifies a string against a provided signature using a specified certificate.

**Returns**
void

**Supported Script Types**
Server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

### Module
N/crypto/certificate Module

### Parent Object
Parameters

### Sibling Object Members
Verifier Object Members

### Since
2019.1
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.signature</td>
<td>string</td>
<td>required</td>
<td>The signature to be verified.</td>
</tr>
<tr>
<td>options.signatureEncoding</td>
<td>string</td>
<td>optional</td>
<td>The signature's encoding in Base64 format.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_SIGNATURE</td>
<td>Signature is not verified. This can occur if the certificate or hash algorithm is not correct in the Verifier object or the signature is not valid for the supplied string.</td>
</tr>
</tbody>
</table>

certificate.createSigner(options)

Method Description

Creates the signer object for signing plain strings.

Returns

A certificate.Signer object

Supported Script Types

Server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

Governance

10 units

Module

N/crypto/certificate Module

Since

2019.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.certId</td>
<td>string</td>
<td>required</td>
<td>The script ID of the digital certificate.</td>
</tr>
<tr>
<td>options.algorithm</td>
<td>string</td>
<td>required</td>
<td>The hash algorithm.</td>
</tr>
</tbody>
</table>

certificate.createVerifier(options)

Method Description

Creates the verifier object for verifying signatures of plain strings.

Returns

A Parameters object

Supported Script Types

Server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

Governance

10 units
Module  |  N/crypto/certificate Module
--- | ---
Since  |  2019.1

## Parameters

![Note: The options parameter is a JavaScript object.](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.certId</td>
<td>string</td>
<td>required</td>
<td>The script ID of the digital certificate.</td>
</tr>
<tr>
<td>options.algorithm</td>
<td>string</td>
<td>required</td>
<td>Hash algorithm</td>
</tr>
</tbody>
</table>

### certificate.verifyXmlSignature(options)

**Method Description**  
Verifies the signature in the signedXml object or string.

**Returns**  
void

**Supported Script Types**  
Server-side scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**  
10 units

**Module**  
N/crypto/certificate Module

**Since**  
2019.1

### Parameters

![Note: The options parameter is a JavaScript object.](image)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.signedXml</td>
<td>string</td>
<td>required</td>
<td>Signed XML.</td>
</tr>
<tr>
<td>options.rootTag</td>
<td>string</td>
<td>required</td>
<td>Signed root XML tag.</td>
</tr>
<tr>
<td>options.certId</td>
<td>string</td>
<td>optional</td>
<td>The script ID for the digital certificate.</td>
</tr>
</tbody>
</table>

### certificate.signXml(options)

**Method Description**  
Signs the inputXml string using the certId.

**Returns**  
certificate.SignedXml

**Supported Script Types**  
Server-side scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**  
10 units
Module | N/crypto/certificate Module
---|---
Since | 2019.1

**Parameters**

> **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.xmlString</td>
<td>string</td>
<td>required</td>
<td>Input XML string</td>
</tr>
<tr>
<td>options.certId</td>
<td>string</td>
<td>required</td>
<td>Certificate ID</td>
</tr>
<tr>
<td>options.algorithm</td>
<td>string</td>
<td>required</td>
<td>Hash algorithm</td>
</tr>
<tr>
<td>options.rootTag</td>
<td>string</td>
<td>required</td>
<td>Root tag of XML section to sign</td>
</tr>
</tbody>
</table>

**certificate.HashAlg**

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>The hash algorithm.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Supported digest methods are SHA1, SHA256, SHA384, and SHA512 for RSA and ECDSA encryption algorithms and SHA1 and SHA256 for DSA.</td>
</tr>
</tbody>
</table>

> **Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
</table>

Module | N/crypto/certificate Module
Sibling Module Members | N/crypto/certificate Module Members
Since | 2019.1

**Values**

- SHA1
- SHA256
- SHA384
- SHA512

**N/currency Module**

Load the N/currency module when you want to work with exchange rates within your NetSuite account. You can use this module to find the exchange rate between two currencies based on a certain date.
To use multiple currencies, the Multiple Currencies feature must be enabled. For information on enabling this feature, see the help topic Enabling the Multiple Currencies Feature.

**Note:** Currency formatting is handled by the `N/format` Module.

- N/currency Module Member
- N/currency Module Script Sample

### N/currency Module Member

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>currency.exchangeRate(options)</code></td>
<td><code>number</code>/<code>string</code></td>
<td>Client and server-side scripts</td>
<td>Returns an exchange rate between two currencies.</td>
</tr>
</tbody>
</table>

### N/currency Module Script Sample

The following example obtains the exchange rate between the Canadian dollar and the US dollar on July 28, 2015.

**Note:** This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/currency'],
    function(currency) {
        function getUSDFromCAD() {
            var canadianAmount = 100;
            var rate = currency.exchangeRate(
                {source: 'CAD',
                target: 'USD',
                date: new Date('7/28/2015')
            });
            var usdAmount = canadianAmount * rate;
            getUSDFromCAD();
        }
    });
}
```

currency.exchangeRate(options)

| Method Description | Method used to return the exchange rate between two currencies based on a certain date. |
The source currency is looked up relative to the target currency on the effective date. For example, if use British pounds for the source and US dollars for the target and the method returns '1.52', this means that if you were to enter an invoice today for a GBP customer in your USD subsidiary, the rate would be 1.52.

The exchange rate values are sourced from the Currency Exchange Rate record.

**Note:** The Currency Exchange Rate record itself is not a scriptable record.

### Returns

The exchange rate as a decimal number

### Supported Script Types

Client and server-side scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

### Governance

10 units

**Module**

N/currency Module

**Since**

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.date</td>
<td>Date</td>
<td>optional</td>
<td>▪ Pass in a new Date object. For example, date: new Date('7/28/2015') ▪ If date is not specified, then it defaults to today (the current date). ▪ The date determines the exchange rate in effect. If there are multiple rates, it is the latest entry on that date. ▪ Use the same date format as your NetSuite account.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.source</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>▪ The internal ID or three-letter ISO code for the currency you are converting from. ▪ For example, you can use either 1 (internal ID) or USD (currency code). ▪ If the Multiple Currencies feature is enabled, from your account, you can view a list of all the currency internal IDs and ISO codes at Lists &gt; Accounting &gt; Currencies.</td>
</tr>
<tr>
<td>options.target</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>▪ The internal ID or three-letter ISO code for the currency you are converting to.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISSING_REQD_ARGUMENT</td>
<td>exchangeRate: Missing a required argument: &lt;source/target&gt;</td>
<td>The source or target argument is missing.</td>
</tr>
</tbody>
</table>
### N/currency Module

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_CURRENCY_ID</td>
<td>You have entered an invalid currency symbol or internal ID: <code>&lt;target/source&gt;</code></td>
<td>The source or target argument is invalid. If the Multiple Currencies feature is enabled, from your account, you can view a list of currency internal IDs and ISO codes at Lists &gt; Accounting &gt; Currencies.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currency Module Script Sample.

```javascript
//Add additional code
...
var canadianAmount = 100;
var rate = currency.exchangeRate({
  source: 'CAD',
  target: 'USD',
  date: new Date('7/28/2015')
});
var usdAmount = canadianAmount * rate;
...
//Add additional code
```

### N/currentRecord Module

You use the N/currentRecord module to access the record that is active in the current client-side context. This module is always a dynamic object and mode of work is always dynamic, not deferred dynamic/standard. For more information, see the help topic SuiteScript 2.0 – Standard and Dynamic Modes. Be aware that when the current record is in view mode it cannot be edited; it is a read-only record when in view mode. As such, any set APIs do not work on the current record in view mode.

You can use the currentRecord module in the following types of scripts:

- **Entry point client scripts** — These scripts use the @NScriptType ClientScript annotation. (For details, see the help topic SuiteScript 2.0 JSDoc Validation.) The system automatically provides this type of script with a currentRecord.CurrentRecord object that represents the current record. For this reason, an entry point client script does not have to explicitly load the currentRecord module. To access the currentRecord object, create a variable and initialize it to the value of the scriptContext.currentRecord property, which is available in each of the SuiteScript 2.0 Client Script Entry Points and API. For an example, see the help topic SuiteScript Client Script Sample.

- **Client-side custom modules** — These scripts do not use an @NScriptType annotation (see the help topic SuiteScript 2.0 Custom Modules). For these scripts, you must manually load the currentRecord module by naming it in the script’s define statement. Additionally, you must actively retrieve a currentRecord.CurrentRecord object by using the currentRecord.get() or currentRecord.get.promise() method. For an example, see N/currentRecord Module Script Sample.

Like the N/record Module, the currentRecord module provides access to body and sublist fields. However, the record module is recommended for server scripts and for cases where a client-side script needs to interact with a record other than the currently active record. By contrast, the currentRecord module is recommended for client-side scripts that need to interact with the currently active record.
Additionally, the functionality of the two modules varies slightly. For example, the currentRecord module does not permit the editing of subrecords, although subrecords can be retrieved in view mode. For additional details, see the following topics:

- N/currentRecord Module Members
- Column Object Members
- CurrentRecord Object Members
- Field Object Members
- Sublist Object Members
- N/currentRecord Module Script Sample

**Note:** SuiteScript supports working with standard NetSuite records and with instances of custom record types. Supported standard record types are described in the SuiteScript Records Browser. Refer also to SuiteScript Supported Records. For help interacting with an instance of a custom record type, see the help topic Custom Record.

### N/currentRecord Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>currentRecord.###Column</td>
<td>Object</td>
<td>Client scripts</td>
<td>Encapsulates a column of a sublist on the current record.</td>
</tr>
<tr>
<td></td>
<td>currentRecord.###CurrentRecord</td>
<td></td>
<td>Client scripts</td>
<td>Represents the record active on the current page.</td>
</tr>
<tr>
<td></td>
<td>currentRecord.Field</td>
<td>Object</td>
<td>Client scripts</td>
<td>Represents a body or sublist field.</td>
</tr>
<tr>
<td></td>
<td>currentRecord.Sublist</td>
<td>Object</td>
<td>Client scripts</td>
<td>Represents a sublist.</td>
</tr>
<tr>
<td>Method</td>
<td>currentRecord.get()</td>
<td>currentRecord.###CurrentRecord</td>
<td>Client scripts</td>
<td>Retrieves a record object that represents the current record.</td>
</tr>
<tr>
<td></td>
<td>currentRecord.###get.###promise()</td>
<td></td>
<td>Client scripts</td>
<td>Retrieves a promise for an object that represents the current record.</td>
</tr>
</tbody>
</table>

### Column Object Members

The following members are called on the currentRecord.Column object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Column.id</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the internal ID of the column.</td>
</tr>
<tr>
<td></td>
<td>Column.label</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the UI label for the column.</td>
</tr>
<tr>
<td></td>
<td>Column.sublistId</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the internal ID of the standard or custom sublist that contains the column.</td>
</tr>
<tr>
<td></td>
<td>Column.type</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the column type.</td>
</tr>
</tbody>
</table>
CurrentRecord Object Members

The following members are called on the `currentRecord.CurrentRecord` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>CurrentRecord.cancelLine(options)</code></td>
<td><code>currentRecord.CurrentRecord</code></td>
<td>Client scripts</td>
<td>Cancels the changes made to the currently selected line.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.commitLine(options)</code></td>
<td><code>currentRecord.CurrentRecord</code></td>
<td>Client scripts</td>
<td>Commits the currently selected line.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.findMatrixSublistLineWithValue(options)</code></td>
<td><code>number</code></td>
<td>Client scripts</td>
<td>Returns the line number of the first line that contains the specified value in the matrix column.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.findSublistLineWithValue(options)</code></td>
<td><code>number</code></td>
<td>Client scripts</td>
<td>Gets the line number for the first occurrence of a field value in a sublist.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getLineCount(options)</code></td>
<td><code>number</code></td>
<td>Client scripts</td>
<td>Returns the number of lines in the sublist.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getMatrixHeaderCount(options)</code></td>
<td><code>number</code></td>
<td>Client scripts</td>
<td>Returns the number of columns for the specified matrix.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getMatrixHeaderField(options)</code></td>
<td><code>currentRecord.Field</code></td>
<td>Client scripts</td>
<td>Gets the field for the specified header in the matrix.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getMatrixHeaderValue(options)</code></td>
<td>`number</td>
<td>Date</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getMatrixSublistField(options)</code></td>
<td><code>currentRecord.Field</code></td>
<td>Client scripts</td>
<td>Gets the field for the specified sublist in the matrix.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getMatrixSublistValue(options)</code></td>
<td>`number</td>
<td>Date</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getField(options)</code></td>
<td><code>currentRecord.Field</code></td>
<td>Client scripts</td>
<td>Gets a field object from the record.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getLineCount(options)</code></td>
<td><code>number</code></td>
<td>Client scripts</td>
<td>Returns the number of lines in the sublist.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getHeaderCount(options)</code></td>
<td><code>number</code></td>
<td>Client scripts</td>
<td>Returns the number of columns for the specified matrix.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getHeaderField(options)</code></td>
<td><code>currentRecord.Field</code></td>
<td>Client scripts</td>
<td>Gets the field for the specified header in the matrix.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getHeaderHeaderValue(options)</code></td>
<td>`number</td>
<td>Date</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getSublistField(options)</code></td>
<td><code>currentRecord.Field</code></td>
<td>Client scripts</td>
<td>Gets the specified field object from the sublist.</td>
</tr>
<tr>
<td></td>
<td><code>CurrentRecord.getSublistValue(options)</code></td>
<td><code>string</code></td>
<td>Client scripts</td>
<td>Gets the value of the field in a sublist by a string representation.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>getSublistValue</td>
<td>number</td>
<td>Date</td>
<td>string</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>getSubrecord</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>getText</td>
<td>string</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>getValue</td>
<td>number</td>
<td>Date</td>
<td>string</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>hasSublistSubrecord</td>
<td>boolean</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>hasSubrecord</td>
<td>boolean</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>insertLine</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>removeSublistSubrecord</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>removeLine</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>removeSubrecord</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>selectLine</td>
<td>void</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>selectNewLine</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>setCurrentSublistValue</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>setCurrentSublistText</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>setCurrentSublistValue</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>setCurrentSublistValue</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>setCurrentSublistValue</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>setText</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>CurrentRecord.</td>
<td>setValue</td>
<td>CurrentRecord</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>Property</td>
<td>CurrentRecord.id</td>
<td>number (read-only)</td>
<td>Client scripts</td>
<td>Returns the internal record ID.</td>
</tr>
<tr>
<td>Property</td>
<td>CurrentRecord.isDynamic</td>
<td>boolean true</td>
<td>false (read-only)</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Property</td>
<td>CurrentRecord.type</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the record type.</td>
</tr>
</tbody>
</table>
Field Object Members

The following members are called on the `currentRecord.Field` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Field.getSelectOptions()</td>
<td>array</td>
<td>Client scripts</td>
<td>Returns an array of available options on a standard or custom select, multiselect, or radio field as key-value pairs. Only the first 1,000 available options are returned.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.insertSelectOption()</td>
<td>void</td>
<td>Client scripts</td>
<td>Inserts an option into certain types of select and multiselect fields. This method is usable only in fields that were added by a front-end Suitelet or beforeLoad user event script.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.removeSelectOption()</td>
<td>void</td>
<td>Client scripts</td>
<td>Removes an option from certain types of select and multiselect fields. This method is usable only in fields that were added by a front-end Suitelet or beforeLoad user event script. It is supported only in client scripts.</td>
</tr>
<tr>
<td>Object</td>
<td>Field.id</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the internal ID of a standard or custom body or sublist field.</td>
</tr>
<tr>
<td></td>
<td>Field.isDisabled</td>
<td>boolean true</td>
<td>Client scripts</td>
<td>Returns <code>true</code> if the standard or custom field is disabled on the record form, or <code>false</code> otherwise.</td>
</tr>
<tr>
<td></td>
<td>Field.isDisplay</td>
<td>boolean true</td>
<td>Client scripts</td>
<td>Returns <code>true</code> if the field is set to display on the record form, or <code>false</code> otherwise. This property is read-only for sublist fields.</td>
</tr>
<tr>
<td></td>
<td>Field.isMandatory</td>
<td>boolean true</td>
<td>Client scripts</td>
<td>Returns <code>true</code> if the standard or custom field is mandatory on the record form, or <code>false</code> otherwise.</td>
</tr>
<tr>
<td></td>
<td>Field.isPopup</td>
<td>boolean true</td>
<td>Client scripts</td>
<td>Returns <code>true</code> if the field is a popup list field, or <code>false</code> otherwise.</td>
</tr>
<tr>
<td></td>
<td>Field.isReadOnly</td>
<td>boolean true</td>
<td>Client scripts</td>
<td>Returns <code>true</code> if the field on the record form cannot be edited, or <code>false</code> otherwise. For textarea fields, this property can be read or written to. For all other fields, this property is read-only.</td>
</tr>
</tbody>
</table>
### Field isVisible

- **Member Type:** Field
- **Name:** isVisible
- **Return Type / Value Type:** boolean (true | false) (read-only)
- **Supported Script Types:** Client scripts
- **Description:** Returns true if the field is visible on the record form, or false otherwise.

### Field label

- **Member Type:** Field
- **Name:** label
- **Return Type / Value Type:** string (read-only)
- **Supported Script Types:** Client scripts
- **Description:** Returns the UI label for a standard or custom field body or sublist field.

### Field sublistId

- **Member Type:** Field
- **Name:** sublistId
- **Return Type / Value Type:** string (read-only)
- **Supported Script Types:** Client scripts
- **Description:** Returns the ID of the sublist associated with the specified sublist field.

### Field type

- **Member Type:** Field
- **Name:** type
- **Return Type / Value Type:** string (read-only)
- **Supported Script Types:** Client scripts
- **Description:** Returns the type of a body or sublist field.

### Sublist Object Members

The following members are called on the currentRecord.Sublist object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Sublist.getColumn(options)</td>
<td>currentRecord.Column</td>
<td>Client scripts</td>
<td>Returns a column in the sublist.</td>
</tr>
<tr>
<td>Property</td>
<td>Sublist.id</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the internal ID of the sublist.</td>
</tr>
<tr>
<td></td>
<td>Sublist.isChanged</td>
<td>boolean (true</td>
<td>Client scripts</td>
<td>Indicates whether the sublist has changed on the current record form.</td>
</tr>
<tr>
<td></td>
<td>Sublist.isDisplay</td>
<td>boolean (true</td>
<td>Client scripts</td>
<td>Indicates whether the sublist is displayed on the current record form.</td>
</tr>
<tr>
<td></td>
<td>Sublist.type</td>
<td>string (read-only)</td>
<td>Client scripts</td>
<td>Returns the sublist type.</td>
</tr>
</tbody>
</table>

### N/currentRecord Module Script Sample

The following example is a custom module client script named clientDemo.js. This script updates fields on the current record. After you upload clientDemo.js to a NetSuite account, it can be called by other scripts, as shown in the subsequent sample.

Because clientDemo.js is a custom module script, it must manually load the currentRecord module by naming it in the define statement. Additionally, it must actively retrieve a CurrentRecord object. It does so by using the currentRecord.get() method.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

```javascript
/**
 * @NApiVersion 2.0
 */
define(['N/currentRecord'], function(currentRecord) {
  return {
    test_set_getValue: function() {
      // Code here
    }
  }
})
```
The following example is a user event script deployed on a non-inventory item record. Before the record loads, the script updates the form used by the record to add new text fields, a sublist, and buttons that call the clientDemo.js methods. The buttons access the current record and set values for some of the form's fields. The use case for this example is to set up a page, adding fields and buttons, so that you can use the code you made in the first example, and see the fields and buttons in action.

**Note:** This sample script uses the `define` function. Note that you cannot use On Demand Debugging to step through a `define` function. You must use Deployed Debugging to step through this script.
currentRecord.Column

Object Description: Encapsulates a column of a sublist on the current record.

For a complete list of this object's properties, see Column Object Members.

Supported Script Types: Client scripts
## N/currentRecord Module

### SuiteScript 2.0 API Reference

Since 2016.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code ...
var objColumn = objSublist.getColumn({
  fieldId: 'item'
});
...
//Add additional code
```

### Column.id

**Property Description:** Returns the internal ID of the column.

**Type:** string (read-only)

**Supported Script Types:** Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

### Column.label

**Property Description:** Returns the internal ID of the column.

**Type:** string (read-only)

**Supported Script Types:** Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
var columnlabel = objColumn.label;
...
//Add additional code
```

### Column.sublistId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the internal ID of the standard or custom sublist that contains the column.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client scripts  
For more information, see the help topic [SuiteScript 2.0 Client Script Type](#). |

### Column.type

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the column type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client scripts  
For more information, see the help topic [SuiteScript 2.0 Client Script Type](#). |
Since 2016.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var columntype = objColumn.type;
...
//Add additional code
```

currentRecord.CurrentRecord

**Object Description**
Encapsulates the record active on the current page.

**Supported Script Types**
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Module**
N/currentRecord Module

**Since**
2016.2

Syntax

⚠️ **Important:** The following code snippets show the syntax for this member. These snippets are not a functional examples. For a complete script example, see N/currentRecord Module Script Sample and SuiteScript Client Script Sample.

The following snippet shows the retrieval of a currentRecord object in a custom module where the currentRecord was explicitly loaded.

```javascript
//Add additional code
...
var objRecord = currentRecord.get();
...
//Add additional code
```

In an entry point client script, you do not have to use the get method to retrieve the current record. (An entry point client script is one that uses the `@NScriptType ClientScript` annotation.) In these scripts, a currentRecord object is automatically created when the script is loaded. It is part of the context object that passed to each of the client script type's entry points. However, you do have to create a variable to represent the current record, as shown in the following snippet.

```javascript
//Add additional code
...
function pageInit(context) {
    var currentRec = context.currentRecord;
    ...
//Add additional code
```
CurrentRecord.cancelLine(options)

Method Description  Cancels the currently selected line on a sublist.

Returns  The currentRecord.CurrentRecord object that called the method.

Supported Script Types  Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance  None

Module  N/currentRecord Module

Since  2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
objRecord.cancelLine({
  sublistId: 'item'
});
...
//Add additional code
```

CurrentRecord.commitLine(options)

Method Description  Commits the currently selected line on a sublist.
Returns

The `currentRecord.CurrentRecord` object that called the method.

Supported Script Types

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance

None

Module

N/currentRecord Module

Since

2016.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
objRecord.commitLine({
    sublistId: 'item'
});
...
//Add additional code
```

**CurrentRecord.findMatrixSublistLineWithValue(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the line number of the first instance where a specified value is found in a specified column of the matrix.</td>
<td>number</td>
<td>Client scripts</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The ID of the matrix field. See the help topic How do I find a field’s internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>required</td>
<td>The value to search for.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number of the field.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var lineNumber = objRecord.findMatrixSublistLineWithValue({
    sublistId: 'item'
});
...
//Add additional code
```

**CurrentRecord.findSublistLineWithValue(options)**

**Method Description**

Returns the line number for the first occurrence of a field value in a sublist.
Returns

A line number as a number, or -1 if not found.

Supported Script Types

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance

None

Module

N/currentRecord Module

Since

2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>Date</td>
<td>string</td>
<td>array</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or not defined.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var lineNumber = objRecord.findSublistLineWithValue({
  sublistId: 'item',
  fieldId: 'item',
  value: 233
});
...
```
CurrentRecord.getCurrentMatrixSublistValue(options)

Method Description
Gets the value for the currently selected line in the matrix.
Gets a numeric value for rate and ratehighprecision fields.

Returns
number | Date | string | array | boolean true | false

Supported Script Types
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance
None

Module
N/currentRecord Module

Since
2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the matrix field.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
```
```javascript
var matrixValue = objRecord.getCurrentMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12
});
...
// Add additional code
```

### `CurrentRecord.getCurrentSublistIndex(options)`

**Method Description**

Returns the line number of the currently selected line.

**Returns**

number

**Supported Script Types**

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**

None

**Module**

`N/currentRecord Module`

**Since**

2016.2

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information, see the help topic</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/currentRecord Module Script Sample`.

```javascript
// Add additional code
...
var currIndex = objRecord.getCurrentSublistIndex({
    sublistId: 'item'
});
```
CurrentRecord.getCurrentSublistSubrecord(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the subrecord for the associated sublist field on the current line. The subrecord object is retrieved in view mode.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>currentRecord.CurrentRecord</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client scripts  
For more information, see the help topic SuiteScript 2.0 Client Script Type. |
| Governance        | None                                                                                                          |
| Module             | N/currentRecord Module                                                                                         |
| Since              | 2016.2                                                                                                        |

### Parameters

**Note:** The options parameter is a JavaScript object. If no subrecord instance exists, the system creates one. For more information, see the help topic Subrecord Scripting in SuiteScript 2.0 Compared With 1.0.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
//Add additional code
var objSubrecord = objRecord.getCurrentSublistSubrecord({
  sublistId: 'item',
  fieldId: 'item'
});
...
//Add additional code
```
CurrentRecord.getCurrentSublistText(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns string</td>
<td></td>
<td>Client scripts</td>
</tr>
<tr>
<td>Note: For multiselect fields, returns an array.</td>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
<tr>
<td>Governance None</td>
<td>Module N/currentRecord Module</td>
<td></td>
</tr>
<tr>
<td>Since 2016.2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See the help topic How do I find a field’s internal ID?</td>
<td></td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var fieldName = objRecord.getCurrentSublistText({
    sublistId: 'item',
    fieldId: 'item'
});
```
CurrentRecordgetCurrentSublistValue(options)

Method Description
Returns the value of a sublist field on the currently selected sublist line.

Returns
number | Date | string | array | boolean true | false

Supported Script Types
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance
None

Module
N/currentRecord Module

Since
2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...```

SuiteScript 2.0 API Reference
CurrentRecord.getField(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a field object from a record.</td>
<td>Client scripts</td>
<td>None</td>
<td>N/currentRecord Module</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td>currentRecord.Field</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameters</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See the help topic How do I find a field's internal ID?</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

```
//Add additional code
...
var objField = objRecord.getField({
    fieldId: 'item'
});
...
//Add additional code
```
CurrentRecord.getLineCount(options)

**Method Description**
Returns the number of lines in a sublist.

**Returns**
number

**Supported Script Types**
Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**
None

**Module**
N/currentRecord Module

**Since**
2016.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var numLines = objRecord.getLineCount({
  sublistId: 'item'
});
...
//Add additional code
```

CurrentRecord.getMatrixHeaderCount(options)

**Method Description**
Returns the number of columns for the specified matrix.

**Returns**
number

**Supported Script Types**
Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**
None

**Module**
N/currentRecord Module
**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>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var numLines = objRecord.getMatrixHeaderCount({
  sublistId: 'item',
  fieldId: 'item'
});
...
//Add additional code
```

**CurrentRecord.getMatrixHeaderField(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the field for the specified header in the matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>currentRecord.Field</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
</tbody>
</table>
Since 2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var objField = objRecord.getMatrixHeaderField({
    sublistId: 'item',
    fieldId: 'item',
    column: 12
});
//Add additional code
```

CurrentRecord.getMatrixHeaderValue(options)

Method Description
Gets the value for the associated header in the matrix.

Returns
number | Date | string | array | boolean | true | false

Supported Script Types
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See the help topic How do I find a field's internal ID?</td>
<td></td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var value = objRecord.getMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12
});
//Add additional code
```

**CurrentRecord.getMatrixSublistField(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the field for the specified sublist in the matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>currentRecord.Field</td>
</tr>
</tbody>
</table>
Supported Script Types
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance
None

Module
N/currentRecord Module

Since
2016.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See the help topic How do I find a field's internal ID?</td>
<td></td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var objField = objRecord.getMatrixSublistField( {
  sublistId: 'item',
  fieldId: 'item',
  column: 12,
  line: 3
});
//Add additional code
```
CurrentRecord.getMatrixSublistValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the value for the associated field in the matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var value = objRecord.getMatrixSublistValue({
  sublistId: 'item',
  fieldId: 'item',
...});
```
CurrentRecord.getSublist(options)

Method Description | Returns the specified sublist.
--- | ---

Returns | currentRecord.Sublist

Supported Script Types | Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance | None

Module | N/currentRecord Module

Since | 2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var objSublist = objRecord.getSublist({
   sublistId: 'item'
});
...
//Add additional code
```

CurrentRecord.getSublistField(options)

Method Description | Returns a field object from a sublist.
Returns  currentRecord.Field

Supported Script Types  Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance  None

Module  N/currentRecord Module

Since  2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var objField = objRecord.getSublistField({
  sublistId: 'item',
  fieldId: 'item',
  line: 3
});
...
//Add additional code
```
CurrentRecord.getSublistText(options)

Method Description
Returns the value of a sublist field in a text representation. Gets a string value with a "%" for rate and ratehighprecision fields.

Returns
string

Note: For multiselect fields, returns an array.

Supported Script Types
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance
None

Module
N/currentRecord Module

Since
2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
```
CurrentRecord.getSublistValue(options)

Method Description
Returns the value of a sublist field.

Gets a numeric value for rate and ratehighprecision fields.

Returns
number | Date | string | array | boolean true | false

Supported Script Types
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance
None

Module
N/currentRecord Module

Since
2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
var sublistFieldValue = objRecord.getSublistValue({
  sublistId: 'item',
  fieldId: 'item',
  line: 3
});
...
//Add additional code
```

**CurrentRecord.getSubrecord(options)**

### Method Description

Gets the subrecord associated with the field. The subrecord object is available in view mode.

### Returns

`currentRecord.CurrentRecord`

### Supported Script Types

Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

### Governance

None

### Module

N/currentRecord Module

### Since

2016.2

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field. See the help topic <a href="#">How do I find a field's internal ID?</a></td>
<td>2016.2</td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>FIELD_1 IS NOT A_SUBRECORD_FIELD</td>
<td>The specified field is not a subrecord field.</td>
</tr>
<tr>
<td>FIELD_1 IS DISABLED YOU CANNOT APPLY SUBRECORD FUNCTION ON THIS FIELD</td>
<td>The specified field is disabled.</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
// Add additional code
...
var sublistFieldValue = objRecord.getSubrecord({
  fieldId: 'subrecord'
});
...
// Add additional code
```

### CurrentRecord.getText(options)

**Method Description**

Returns the text representation of a field value.

Gets a string value with a "%" for rate and ratehighprecision fields.

**Returns**

- **string**

  **Note:** For multiselect fields, returns an array.

**Supported Script Types**

- Client scripts

  For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

**Governance**

- None

**Module**

- N/currentRecord Module

**Since**

- 2016.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See the help topic <a href="#">How do I find a field's internal ID?</a></td>
<td></td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>
Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var fieldidname = objRecord.getText({
   fieldId: 'item'
});
...
//Add additional code
```

CurrentRecord.getValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the value of a field.</td>
<td>number</td>
<td>Client scripts</td>
<td>None</td>
<td>N/currentRecord Module</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Returns

number | Date | string | array | boolean true | false

Supported Script Types

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance

None

Module

N/currentRecord Module

Since

2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
```
CurrentRecord.hasCurrentSublistSubrecord(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a value indicating whether the associated sublist field has a subrecord on the current line. This method can only be used on dynamic records.</td>
<td>boolean true</td>
<td>Client scripts</td>
<td>None</td>
<td>N/currentRecord Module</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a subrecord. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
var hasSubrecord = objRecord.hasCurrentSublistSubrecord({
  sublistId: 'item',
  fieldId: 'item'
});
```
CurrentRecord.hasSublistSubrecord(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a value indicating whether the associated sublist field contains a subrecord.</td>
<td>boolean true</td>
<td>Client scripts</td>
<td>None</td>
<td>N/currentRecord Module</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a subrecord. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var hasSubrecord = objRecord.hasSublistSubrecord({
  sublistId: 'item',
  fieldId: 'item',
  line: 3
});
...
//Add additional code
```
CurrentRecord.hasSubrecord(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns a value indicating whether the field contains a subrecord.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>boolean true</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the field that may contain a subrecord.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See the help topic <a href="#">How do I find a field's internal ID?</a></td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
var hasSubrecord = objRecord.hasSubrecord({
   fieldId: 'item'
});
...
//Add additional code
```

CurrentRecord.insertLine(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Inserts a sublist line.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>currentRecord.CurrentRecord</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
</tbody>
</table>
Since 2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number to insert. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.ignoreRecalc</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, scripting recalculation is ignored. The default value is false.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
objRecord.insertLine({
  sublistId: 'item',
  line: 3,
  ignoreRecalc: true
});
...
//Add additional code
```

CurrentRecord.removeCurrentSublistSubrecord(options)

Method Description

Removes the subrecord for the associated sublist field.

Returns

currentRecord.CurrentRecord

Supported Script Types

Client scripts
For more information, see the help topic *SuiteScript 2.0 Client Script Type.*

<table>
<thead>
<tr>
<th>Governance</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td><em>N/currentRecord Module</em></td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic <em>Using the SuiteScript Records Browser.</em></td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic <em>How do I find a field's internal ID?</em></td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see *N/currentRecord Module Script Sample.*

```javascript
//Add additional code
...
objRecord.removeCurrentSublistSubrecord({
  sublistId: 'item',
  fieldId: 'item'
});
...
//Add additional code
```

**CurrentRecord.removeLine(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Removes a sublist line.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>currentRecord.CurrentRecord</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>For more information, see the help topic <em>SuiteScript 2.0 Client Script Type.</em></td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td><em>N/currentRecord Module</em></td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number of the sublist to remove. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.ignoreRecalc</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, scripting recalculation is ignored. The default value is false.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
objRecord.removeSubrecord(
  {sublistId: 'item',
   line: 3,
   ignoreRecalc: true
  });
...//Add additional code
```

### CurrentRecord.removeSubrecord(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Removes the subrecord for the associated field.</td>
<td>currentRecord.CurrentRecord</td>
<td>Client scripts</td>
</tr>
</tbody>
</table>

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).
CurrentRecord.selectLine(options)

- **Method Description**: Selects an existing line in a sublist.
- **Returns**: `currentRecord.CurrentRecord`
- **Supported Script Types**: Client scripts
  - For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).
- **Governance**: None
- **Module**: `N/currentRecord Module`
- **Since**: 2016.2

### Parameters

- **Note**: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>
CurrentRecord.selectNewLine(options)

**Method Description**
Selects a new line at the end of a sublist.

**Returns**
currentRecord.CurrentRecord

**Supported Script Types**
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**
None

**Module**
N/currentRecord Module

**Since**
2016.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
var lineNum = objRecord.selectLine(
    sublistId: 'item',
    line: 3
);
...
// Add additional code
```

## Important:
The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.
**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>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
objRecord.selectNewLine({
  sublistId: 'item'
});
...
// Add additional code
```

**CurrentRecord.setCurrentMatrixSublistValue(options)**

**Method Description**

Sets the value for the line currently selected in the matrix.

Sets a numeric value for rate and ratehighprecision fields.

This method is not available for standard records.

**Returns**

`currentRecord.CurrentRecord`

**Supported Script Types**

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**

None

**Module**

N/currentRecord Module

**Since**

2016.2

**Parameters**

Note: The options parameter is a JavaScript object.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See the help topic How do I find a field’s internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>Date</td>
<td>string</td>
<td>array</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Text, Radio and Select fields accept string values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Checkbox fields accept Boolean values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Date and DateTime fields accept Date values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Integer, Float, Currency and Percent fields accept number values.</td>
<td></td>
</tr>
<tr>
<td>options.ignoreFieldChange</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, the field change and slaving event is ignored. By default, this value is false.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.forceSyncSourcing</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether to perform field sourcing synchronously. If set to true, sources dependent field information for empty fields synchronously. Defaults to false - dependent field values are not sourced synchronously.</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
```
```javascript
objRecord.setCurrentMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
//Add additional code
```

---

**CurrentRecord.setCurrentSublistText(options)**

| Method Description | Sets the value for the field in the currently selected line by a text representation. Sets a string value with a "%" for rate and ratehighprecision fields. |
| Supported Script Types | Client scripts |
| Returns | currentRecord.CurrentRecord |
| Governance | None |
| Module | N/currentRecord Module |
| Since | 2016.2 |

### Parameters

> **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.text</td>
<td>string</td>
<td>required</td>
<td>The text to set the value to.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.#ignore#Field#Change</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, the field change and slaving event is ignored. By default, this value is false.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.#force#Sync#Sourcing</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether to perform field sourcing synchronously.</td>
<td>2019.1</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If set to <code>true</code>, sources dependent field information for empty fields synchronously. Defaults to <code>false</code> - dependent field values are not sourced synchronously.</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>A_USER_SCRIPT_IS_ATTEMPTING_TO_EDIT_A_READONLY_SUBLIST_THIS_SUBLIST_IS_CURRENTLY_IN_READ_MODE</td>
<td>A user tries to edit a read-only sublist field.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
objRecord.setCurrentSublistText({
  sublistId: 'item',
  fieldId: 'item',
  text: 'value',
  ignoreFieldChange: true,
  forceSyncSourcing: true
});
...
//Add additional code
```

**CurrentRecord.setCurrentSublistValue(options)**

**Method Description**
Sets the value for the field in the currently selected line.

**Important:** When you edit a sublist line with SuiteScript, it triggers an internal validation of the sublist line. If the line validation fails, the script also fails. For example, if your script edits a closed catch up period, the validation fails and prevents SuiteScript from editing the closed catch up period.

**Returns**
Sets a numeric value for rate and ratehighprecision fields.

**Supported Script Types**
Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

**Governance**
None
Module | N/currentRecord Module
---|---
Since | 2016.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
</tbody>
</table>
| options.value    | number | Date | string | array | boolean | true | false | required | The value to set the field to. The value type must correspond to the field type being set. For example:
  - Text, Radio and Select fields accept string values.
  - Checkbox fields accept Boolean values.
  - Date and DateTime fields accept Date values.
  - Integer, Float, Currency and Percent fields accept number values. | 2016.2|
| options.ignoreFieldChange | boolean | true | false | optional | If set to true, the field change and slaving event is ignored. By default, this value is false. | 2016.2|
| options.forceSyncSourcing | boolean | true | false | optional | Indicates whether to perform field sourcing synchronously. By default, this value is false. | 2019.1|

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>A_SCRIPT_IS_ATTEMPTING_TO_EDIT</td>
<td>A user tries to edit a read-only sublist field.</td>
</tr>
</tbody>
</table>
## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
objRecord.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true
});
...
//Add additional code
```

### CurrentRecord.setMatrixHeaderValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Sets the value for the associated header in the matrix. Sets a numeric value for rate and ratehighprecision fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>currentRecord.CurrentRecord</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See the help topic How do I find a field's internal ID?</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2016.2</td>
</tr>
</tbody>
</table>

SuiteScript 2.0 API Reference
### Parameter Types

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
</table>
| options.value      | number | Date | string | array | boolean | true | false | The value to set the field to. The value type must correspond to the field type being set. For example:  
- Text, Radio and Select fields accept string values.  
- Checkbox fields accept Boolean values.  
- Date and DateTime fields accept Date values.  
- Integer, Float, Currency and Percent fields accept number values. | 2016.2 |
| options.**ignoreFieldChange** | boolean | true | false | If set to true, the field change and slaving event is ignored. By default, this value is false. | 2016.2 |
| options.**forceSyncSourcing** | boolean | true | false | Indicates whether to perform field sourcing synchronously. If set to true, sources dependent field information for empty fields synchronously. Defaults to false – dependent field values are not sourced synchronously. | 2019.1 |

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
objRecord.setMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
```

---

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see **N/currentRecord Module Script Sample.**
CurrentRecord.setMatrixSublistValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Sets the value for the associated field in the matrix. Sets a numeric value for rate and ratehighprecision fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>currentRecord.CurrentRecord</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

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>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>
| options.value | number | Date | string | array | boolean | true | false | required | The value to set the field to. The value type must correspond to the field type being set. For example:  
  - Text, Radio and Select fields accept string values.  
  - Checkbox fields accept Boolean values.  
  - Date and DateTime fields accept Date values.  
  - Integer, Float, Currency and Percent fields accept number values. | 2016.2 |

Note: The options parameter is a JavaScript object.
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The <code>options.value</code> type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
objRecord.setMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3,
    value: true
});
...
//Add additional code
```

**CurrentRecord.setText(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Sets the value of the field by a text representation. Sets a string value with a &quot;%&quot; for rate and ratehighprecision fields.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th><code>currentRecord.CurrentRecord</code></th>
</tr>
</thead>
</table>

**Supported Script Types**

Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

**Governance**

None

**Module**

N/currentRecord Module

**Since**

2016.2

Parameters

**Note:** The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>
### Parameter Details

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.text</td>
<td>string</td>
<td>required</td>
<td>The text to change the field value to.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.ignoreFieldChange</td>
<td>boolean</td>
<td>true</td>
<td>If set to true, the field change and slaving event is ignored. By default, this value is false.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.forceSyncSourcing</td>
<td>boolean</td>
<td>true</td>
<td>Indicates whether to perform field sourcing synchronously. If set to true, sources dependent field information for empty fields synchronously. Defaults to false – dependent field values are not sourced synchronously.</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
objRecord.setText({
  fieldId: 'item',
  text: 'value',
  ignoreFieldChange: true,
  forceSyncSourcing: true,
});
...
//Add additional code
```

### Method Description

**CurrentRecord.setValue(options)**

- **Sets the value of a field.**
- **Sets a numeric value for rate and ratehighprecision fields.**

### Returns

currentRecord.CurrentRecord

### Supported Script Types

Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

## Governance

<table>
<thead>
<tr>
<th>Module</th>
<th>N/currentRecord Module</th>
</tr>
</thead>
</table>

## Since

2016.2

## Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
</tbody>
</table>
| options.value          | number | Date | string | array | boolean | true | false | required | The value to set the field to. The value type must correspond to the field type being set. For example:  

- Text, Radio, Select and Multi-Select fields accept string values.  
- Checkbox fields accept Boolean values.  
- Date and DateTime fields accept Date values.  
- Integer, Float, Currency and Percent fields accept number values. | 2016.2 |

| options.\#ignore\#Field\#Change | boolean | true | false | optional | If set to true, the field change and slaving event is ignored. By default, this value is false.                                                                                                           | 2016.2 |

| options.\#force\#Sync\#Sourcing | boolean | true | false | optional | Indicates whether to perform field sourcing synchronously. If set to true, sources dependent field information for empty fields synchronously. Defaults to false – dependent field values are not sourced synchronously. | 2019.1 |

## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>Error Code</td>
<td>Thrown If</td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
objRecord.setValue(
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true,
    forceSyncSourcing: true
);
...
//Add additional code
```

### CurrentRecord.id

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The internal ID of a specific record.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var recordid = record.id;
...
//Add additional code
```

### CurrentRecord.isDynamic

| Property Description | Indicates whether the record is in dynamic mode. For more information, see the help topic SuiteScript 2.0 – Standard and Dynamic Modes. This value is set when the record is created or accessed. |

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var recordid = record.id;
...
//Add additional code
```
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
if (record.isDynamic) {
  ...
}
...
//Add additional code
```

### CurrentRecord.type

**Property Description**
The current record's type.

Note the following:
- On an instance of a standard record type, this property is represented by a value from the `record.Type` enum.
- On an instance of a custom record type, this value is populated by the custom record type's string ID. For help finding this ID, see the help topic [Custom Record](#).

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
var recordtype = currentRecord.type;
...
//Add additional code
```
currentRecord.Field

Object Description
Encapsulates a body or sublist field on the current record.

Use the following methods to access the Field object:

- `CurrentRecord.getField(options)`
- `CurrentRecord.getSublistField(options)`

For a complete list of this object's methods and properties, see `N/currentRecord Module`.

Supported Script Types
Client scripts
For more information, see the help topic `SuiteScript 2.0 Client Script Type`.

Module
`N/currentRecord Module`

Since
2016.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/currentRecord Module Script Sample`.

```javascript
//Add additional code
...
var currentRecordField = currentRecord.getField({
   fieldId: 'entity'
});
...
//Add additional code
}
```

Field.getSelectOptions(options)

**Method Description**
Returns an array of available options on a standard or custom select, multiselect, or radio field as key-value pairs.

⚠️ **Important:** You can use this method only in dynamic mode. For additional information on dynamic mode, see `CurrentRecord.isDynamic`.

**Returns**
array

Only the first 1,000 available options are returned in an array.

If there are more than 1,000 available options, an empty array `[]` is returned.

This function returns an array in the following format:

```
[{value: 5, text: 'abc'}, {value: 6, text: '123'}]
```

This function returns `TypeError` if the field is not a supported field for this method.

**Governance**
None
Supported Script Types | Client scripts
--- | ---
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Module | N/currentRecord Module
--- | ---
Since | 2016.2

Parameters

> **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.filter</td>
<td>string</td>
<td>Required</td>
<td>The search string to filter the select options that are returned.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.operator</td>
<td>string</td>
<td>Required</td>
<td>The following operators are supported:</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓ contains (default)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓ is</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>✓startswith</td>
<td></td>
</tr>
</tbody>
</table>

> **Note:** Filter values are case insensitive.

Syntax

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var options = objField.getSelections({
    filter : 'C',
    operator : 'startswith'
});
...
//Add additional code
Field.insertSelectOption(options)
```

Field.insertSelectOption(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th></th>
<th></th>
<th>Inserts an option into certain types of select and multiselect fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This method is usable only in select and multiselect fields that were added by a front-end Suitelet or beforeLoad user event script. The IDs for these fields always have a prefix of custpage.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>Void</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Module

N/currentRecord Module

Since

2016.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.value</td>
<td>string</td>
<td>Required</td>
<td>A string, not visible in the UI, that identifies the option.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.text</td>
<td>string</td>
<td>Required</td>
<td>The label that represents the option in the UI.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.isSelected</td>
<td>boolean</td>
<td>Optional</td>
<td>Determines whether the option is selected by default. If not specified, this value defaults to false.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

Error Code          | Thrown If                                                                                   |
---------------------|---------------------------------------------------------------------------------------------|
SSS_INVALID_UI_OBJECT_TYPE | A script attempts to use this method on the wrong type of field. This method can be used only on select and multiselect fields whose IDs begin with the prefix custpage. |

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example.

```javascript
//Add additional code
...

// Instantiate the field. Note that this method is supported only // on fields whose fieldIds have a prefix of custpage.

var field = call.getField({
   fieldId: 'custpage_select1field'
});

// Insert a new option.

field.insertSelectOption({
   value: 'Option1',
   text: 'alpha'
});
...
```
Field.removeSelectOption(options)

**Method Description**
Removes a select option from certain types of select and multiselect fields.

This method is usable only in select fields that were added by a front-end Suitelet or beforeLoad user event script. The IDs for these fields always have a prefix of custpage.

**Returns**
Void

**Supported Script Types**
Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**
None

**Module**
N/currentRecord Module

**Since**
2016.2

**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>options.value</td>
<td>string</td>
<td>Required</td>
<td>A string, not shown in the UI, that identifies the option.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To remove all options from the list, set this field to null, as follows:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>field.removeSelectOption({</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>value: null,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>});</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_UI_OBJECT_TYPE</td>
<td>A script attempts to use this method on the wrong type of field. This method can be used only on select and multiselect fields whose IDs begin with the prefix custpage.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example.

```javascript
//Add additional code
```
// Instantiate the field. Note that this method is supported only
// on fields whose fieldIds have a prefix of custpage.

var field = call.getField({
    fieldId: 'custpage_select1field'
});

// Remove the appropriate option.

field.removeSelectOption({
    value: 'Option2',
});

...
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Since 2016.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var label = objField.label;
...
//Add additional code
```

Field.isMandatory

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns true if the standard or custom field is mandatory on the record form, or false otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean true</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
</tbody>
</table>

Since 2016.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
if (objField.isMandatory) {
    ...
} else {
    ...
}
//Add additional code
```

Field.isDisabled

<table>
<thead>
<tr>
<th>Property Description</th>
<th>This property reflects the display type of a field. A value of true means the field is disabled. A value of false means the field is enabled. Note also:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- If you are working with a body field, you can use this property to change the field's display type.</td>
</tr>
</tbody>
</table>
If you are working with a sublist field, you can set this property to `true` or `false`, but be aware that this action affects the entire sublist column, even though a sublist field is associated with one line.

For both body and sublist fields, you can use `Field.isDisabled` to determine whether the field is disabled or enabled.

### Field.isDisabled

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean <code>true</code></th>
<th><code>false</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
<td></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
if (objField.isDisabled) {
  ...
}
...
//Add additional code
```

### Field.isPopup

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns <code>true</code> if the field is a popup list field, or <code>false</code> otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean <code>true</code></td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
if (objField.isPopup) {
  ...
}
...
//Add additional code
```
## Field.isDisplay

### Property Description
Returns `true` if the field is set to display on the record form, or `false` otherwise. Fields can be a part of a record even if they are not displayed on the record form. This property is read-only for sublist fields.

### Type
`boolean`: `true` | `false`

### Module
`N/currentRecord Module`

### Supported Script Types
Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

### Since
2016.2

### Syntax

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
if (objField.isDisplay) {
    ...
}
...
//Add additional code
```

## Field.isVisible

### Property Description
Returns `true` if the field is visible on the record form, or `false` otherwise.

### Type
`boolean`: `true` | `false` (read-only)

### Module
`N/currentRecord Module`

### Supported Script Types
Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

### Since
2016.2

### Syntax

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
if (objField.isVisible) {
    ...
}
...
```

---

**SuiteScript 2.0 API Reference**
### Field.isReadOnly

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns true if the field on the record form cannot be edited, or false otherwise. For textarea fields, this property can be read or written to. For all other fields, this property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean true</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
if (objField.isReadOnly) {
  ...
}  
...
//Add additional code
```

### Field.sublistId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the sublist ID for the specified sublist field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var myId = field.sublistId;
```
### Field.type

**Property Description**

Returns the type of a body or sublist field.

For example, the value can return `text`, `date`, `currency`, `select`, `checkbox`, and other similar values.

The maximum character limit for select field types is 801.

**Type**

string (read-only)

**Module**

N/currentRecord Module

**Supported Script Types**

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Since**

2016.2

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var type = objField.type;
...
//Add additional code
```

### currentRecord.Sublist

**Object Description**

Encapsulates a sublist on the current record.

For a complete list of this object's methods and properties, see N/currentRecord Module.

**Supported Script Types**

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Module**

N/currentRecord Module

**Since**

2016.2

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var objSublist = currentRecord.getSublist();
```
Sublist.getColumn(options)

**Method Description**
Returns a column in the sublist.

**Returns**
currentRecord.Column

**Supported Script Types**
Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

**Governance**
None

**Module**
N/currentRecord Module

**Since**
2016.2

**Parameters**

> **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the column field in the sublist. See the help topic How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Syntax**

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/currentRecord Module Script Sample](#).

```javascript
//Add additional code
...
var objColumn = objSublist.getColumn({
    fieldId: 'item'
});
...
//Add additional code
```

Sublist.id

**Property Description**
Returns the internal ID of the sublist.

**Type**
string (read-only)

**Module**
N/currentRecord Module

**Supported Script Types**
Client scripts
Sublist.isChanged

**Property Description**
Indicates whether the sublist has changed on the current record form.

**Type**
boolean true | false (read-only)

**Module**
N/currentRecord Module

**Supported Script Types**
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Since**
2016.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var sublistid = objSublist.id;
...
//Add additional code
```

Sublist.isDisplay

**Property Description**
Indicates whether the sublist is displayed on the current record form.

**Type**
boolean true | false (read-only)

**Module**
N/currentRecord Module

**Supported Script Types**
Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Since**
2016.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
if (objSublist.isChanged) {
    ...
}
...
//Add additional code
```
Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
if (objSublist.isDisplay) {
  ...
}
...
//Add additional code
```

### Sublist.type

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Returns the sublist type.</td>
</tr>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/currentRecord Module</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var sublisttype = objSublist.type;
...
//Add additional code
```

### currentRecord.get()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieves</td>
<td>Retrieves a currentRecord object that represents the record active on the current page.</td>
</tr>
<tr>
<td>Returns</td>
<td>currentRecord.CurrentRecord</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
</tbody>
</table>

Governance | None |

Module | N/currentRecord Module |

Since | 2016.2 |
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANNOT_CREATE_RECORD_INSTANCE</td>
<td>The current record page is not scriptable or an error occurred when creating the record object.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var record = currentRecord.get();
...
//Add additional code
```

currentRecord.get.promise()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Retrieves a promise for a currentRecord object that represents the record active on the current page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Promise</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client scripts  
For more information, see the help topic SuiteScript 2.0 Client Script Type. |
| Governance         | None                                                                                           |
| Module             | N/currentRecord Module                                                                          |
| Since              | 2016.2                                                                                         |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>CANNOT_CREATE_RECORD_INSTANCE</td>
<td>The current record page is not scriptable or an error occurred when creating the record instance.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/currentRecord Module Script Sample.

```javascript
//Add additional code
...
var record = currentRecord.get.promise();
...
//Add additional code
```
N/email Module

Load the N/email module when you want to send email messages from within NetSuite. You can use the N/email module to send regular, bulk, and campaign email.

- N/email Module Members
- N/email Module Script Sample

N/email Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>email.send(options)</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Sends email to an individual or group of recipients and receives bounceback notifications.</td>
</tr>
<tr>
<td></td>
<td>email.send().promise(options)</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Sends email asynchronously to an individual or group of recipients and receives bounceback notifications.</td>
</tr>
<tr>
<td></td>
<td>email.sendBulk(options)</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Sends bulk email.</td>
</tr>
<tr>
<td></td>
<td>email.send().promise(options)</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Sends bulk email asynchronously.</td>
</tr>
<tr>
<td></td>
<td>email.sendCampaignEvent(number, options)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Sends lead nurturing campaigns (drip marketing email).</td>
</tr>
<tr>
<td></td>
<td>email.send().campaignEvent(number, options)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Sends lead nurturing campaigns (drip marketing email) asynchronously.</td>
</tr>
</tbody>
</table>

N/email Module Script Sample

The following example sends email with an attachment.

```javascript
/** *
 * @NApiVersion 2.x
 */
require(['N/email', 'N/record', 'N/file'],
  function(email, record, file) {
    function sendEmailWithAttachment() {
      var senderId = -5;
      var recipientEmail = 'notify@myCompany.com';
      var timeStamp = new Date().getUTCMilliseconds();
      var recipient = record.create({
        type: record.Type.CUSTOMER,  // Replace with your actual record type
        isEmail: true,
        email: recipientEmail
      });
      var fileRecord = file.createRecord('N/file', 'N/fileRecord');
      fileRecord.setValue('fileValue', 'example.txt');
      fileRecord.setValue('isEmail', true);
      fileRecord.createAttachment('example.txt');
      var emailOptions = {
        to: recipient.email,
        subject: 'Example Email with Attachment',
        body: 'This is an example email with an attachment.'
      };
      fileRecord.saveAndClose(null);
      email.send(emailOptions, fileRecord);
    }
    sendEmailWithAttachment();
  });
```

Note: This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.
recipient.setValue({
    fieldId: 'subsidiary',
    value: '1'
});
recipient.setValue({
    fieldId: 'companyname',
    value: 'Test Company' + timeStamp
});
recipient.setValue({
    fieldId: 'email',
    value: recipientEmail
});
var recipientId = recipient.save();
var fileObj = file.load({
    id: 88
});
email.send({
    author: senderId,
    recipients: recipientId,
    subject: 'Test Sample Email Module',
    body: 'email body',
    attachments: [fileObj],
    relatedRecords: {
        entityId: recipientId,
        customRecord: {
            id: recordId,
            recordType: recordTypeId // an integer value
        }
    }
});
sendEmailWithAttachment();

**email.send(options)**

**Method Description**
Method used to send transactional email and receive bounceback notifications if the email is not successfully delivered.

A maximum of 10 recipients (recipient + cc + bcc) is allowed.

The total message size (including attachments) must be 15MB or less. The size of individual attachments cannot exceed 10MB.

**Returns**
void

**Supported Script Types**
Client and server-side scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**
20 usage units

**Module**
N/email Module

**Since**
2015.2
## Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
</table>
| options.author    | number                    | required            | ▪ Internal ID of the email sender.  
▪ To find the internal ID of the sender in the UI, go to Lists > Employees.                                                                                                                          | 2015.2 |
| options.recipients| number[] | string[]          | required            | ▪ The internal ID or email address of the recipient.  
▪ For multiple recipients, use an array of internal IDs or email addresses. You can use an array that contains a combination of internal IDs and email addresses.  
▪ A maximum of 10 recipients (recipient + cc + bcc) is allowed. | 2015.2 |
| options.replyTo   | string                    | optional            | ▪ The email address that appears in the reply-to header when an email is sent out.  
▪ You can use either a single external email address or a generic email address created by the Email Capture Plug-in.                                          | 2015.2 |
| options.cc        | number[] | string[]          | optional            | ▪ The internal ID or email address of the secondary recipient to copy.  
▪ For multiple recipients, use an array of internal IDs or email addresses. You can use an array that contains a combination of internal IDs and email addresses.  
▪ A maximum of 10 recipients (recipient + cc + bcc) is allowed. | 2015.2 |
| options.bcc       | number[] | string[]          | optional            | ▪ The internal ID or email address of the recipient to blind copy.  
▪ For multiple recipients, use an array of internal IDs or email addresses. You can use an array that contains a combination of internal IDs and email addresses.  
▪ A maximum of 10 recipients (recipient + cc + bcc) is allowed. | 2015.2 |
| options.subject   | string                    | required            | ▪ Subject of the outgoing message.                                                                                                                                                                          | 2015.2 |
| options.body      | string                    | required            | ▪ Contents of the email  
▪ SuiteScript formats the body of the email in either plain text or HTML. If HTML tags are present, the message is formatted.                                                                          | 2015.2 |
### Parameter | Type | Required / Optional | Description |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>transactionId</strong></td>
<td>number</td>
<td>optional</td>
<td>The Transaction record(s) associated with the Message record. Use for transaction and opportunity record types.</td>
</tr>
<tr>
<td><strong>activityId</strong></td>
<td>number</td>
<td>optional</td>
<td>The Activity record(s) attached to the Email Message record Use for Case and Campaign record types.</td>
</tr>
<tr>
<td><strong>entityId</strong></td>
<td>number</td>
<td>optional</td>
<td>The Entity record(s) attached to the Email Message record Use for all Entity record types (for example, customer, contact)</td>
</tr>
<tr>
<td><strong>customRecord</strong></td>
<td>Object</td>
<td>optional</td>
<td>The custom record(s) attached to the Email Message record</td>
</tr>
</tbody>
</table>

### options.**attachment**
- **file.File**
  - Type: optional
  - Description: The email file attachments.
  - Required: Optional
  - Since: 2015.2

  - **Note:** Supported for server-side scripts only.

### options.**relatedRecords**
- **Object**
  - Type: optional
  - Description: Object that contains key/value pairs to associate the Message record with related records (including custom records).
  - Required: Optional
  - Since: 2015.2

  - **Note:** See the relatedRecords table for more information.

### options.**isInternalOnly**
- **boolean**
  - Type: optional
  - Description: If true, the Message record is not visible to an external Entity (for example, a customer or contact).
  - Required: Optional
  - Since: 2015.2

  - **Note:** The default value is false.

**relatedRecords**
You can associate the sent email with an array of internal records using key/value pairs. There can be multiple related records but only one of each type.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>customRecord.id</td>
<td>number</td>
<td>optional</td>
<td>The instance ID for the custom record attached to the Email Message record.</td>
<td>2015.2</td>
</tr>
<tr>
<td>customRecord.recordType</td>
<td>string</td>
<td>optional</td>
<td>The integer ID for the custom record type attached to the Message record. This ID is shown as part of the record's URL. For example: custrecordentry.nl?rectype=2&amp;id=56</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** If you use this parameter, `customRecord.recordType` is required.

**Note:** If you use this parameter, `customRecord.id` is required.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/email Module Script Sample.

```javascript
//Add additional code
...
var senderId = -5;
var recipientEmail = 'notify@myCompany.com';
var timeStamp = new Date().getUTCMilliseconds();
var recipient = record.create({
    type: record.Type.CUSTOMER,
    isDynamic: true
});
recipient.setValue({
    fieldId: 'subsidiary',
    value: '1'
});
recipient.setValue({
    fieldId: 'companyname',
    value: 'Test Company' + timeStamp
});
recipient.setValue({
    fieldId: 'email',
    value: recipientEmail
});
var recipientId = recipient.save();
var fileObj = file.load({
    id: 88
```

---

**SuiteScript 2.0 API Reference**

**ORACLE**

---
```javascript
email.send({
    author: senderId,
    recipients: recipientId,
    subject: 'Test Sample Email Module',
    body: 'email body',
    attachments: [fileObj],
    relatedRecords: {
        entityId: recipientId,
        customRecord:
            {
                id: recordId,
                recordType: recordTypeId //an integer value
            }
    }
});
```

//Add additional code

### email.send.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send transactional email asynchronously and receive bounceback notifications if the email is not successfully delivered.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Synchronous Version</td>
<td>email.send(options)</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client and server-side scripts  
For additional information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | 20 usage units                                                                                                     |
| Module             | N/email Module                                                                                                     |
| Since              | 2015.2                                                                                                             |

### email.sendBulk(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>This method is used to send bulk email when a bounceback notification is not required.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note</td>
<td>This API normally uses a bulk email server to send messages. If you need to increase the successful delivery rate of an email, use email.send(options) so that a transactional email server is used.</td>
</tr>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client and server-side scripts  
For additional information, see the help topic SuiteScript 2.0 Script Types. |
### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.author</td>
<td>number</td>
<td>required</td>
<td>Internal ID of the email sender.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To find the internal ID of the sender in the UI, go to Lists &gt; Employees.</td>
<td></td>
</tr>
<tr>
<td>options.recipients</td>
<td>number[]</td>
<td>required</td>
<td>The internal ID or email address of the recipient.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>string[]</td>
<td></td>
<td>For multiple recipients, use an array of internal IDs or email addresses.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use an array that contains a combination of internal IDs and email addresses.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A maximum of 10 recipients (recipient + cc + bcc) is allowed.</td>
<td></td>
</tr>
<tr>
<td>options.replyTo</td>
<td>string</td>
<td>optional</td>
<td>The email address that appears in the reply-to header when an email is sent out.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use either a single external email address or a generic email address created by the Email Capture Plug-in.</td>
<td></td>
</tr>
<tr>
<td>options.cc</td>
<td>number[]</td>
<td>optional</td>
<td>The internal ID or email address of the secondary recipient to copy.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>string[]</td>
<td></td>
<td>For multiple recipients, use an array of internal IDs or email addresses.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use an array that contains a combination of internal IDs and email addresses.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A maximum of 10 recipients (recipient + cc + bcc) is allowed.</td>
<td></td>
</tr>
<tr>
<td>options.bcc</td>
<td>number[]</td>
<td>optional</td>
<td>The internal ID or email address of the recipient to blind copy.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>string[]</td>
<td></td>
<td>For multiple recipients, use an array of internal IDs or email addresses.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can use an array that contains a combination of internal IDs and email addresses.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A maximum of 10 recipients (recipient + cc + bcc) is allowed.</td>
<td></td>
</tr>
</tbody>
</table>
### Parameter | Type | Required / Optional | Description | Since
--- | --- | --- | --- | ---
`options.subject` | string | required | Subject of the outgoing message. | 2015.2
`options.body` | string | required | Contents of the email<br> SuiteScript formats the body of the email in either plain text or HTML. If HTML tags are present, the message is formatted as HTML. Otherwise, the message is formatted in plain text.<br> To display XML as plain text, use an HTML<pre> tag around XML content. | 2015.2
`options.attachments` | File | optional | The email file attachments.<br> You can send multiple attachments of any media type<br> An individual attachment must not exceed 10MB and the total message size must be 15MB or less. | 2015.2
`options.relatedRecords` | Object | optional | Object that contains key/value pairs to associate the Message record with related records (including custom records).<br> See the `relatedRecords` table for more information | 2015.2
`options.isInternalOnly` | boolean | true | If true, the Message record is not visible to an external Entity (for example, a customer or contact).<br> The default value is false. | 2015.2

**Note:** Supported for server-side scripts only.

**Note:** The `relatedRecords` parameter is a JavaScript object and the table below lists its properties.

### relatedRecords
You can associate the sent email with an array of internal records using key/value pairs.

### Parameter | Type | Required / Optional | Description | Since
--- | --- | --- | --- | ---
`transactionId` | number | optional | The Transaction record(s) attached to the Message record<br> Use for transaction and opportunity record types. | 2015.2
`activityId` | number | optional | The Activity record(s) attached to the Email Message record<br> Use for Case and Campaign record types. | 2015.2
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>entityId</td>
<td>number</td>
<td>optional</td>
<td>The Entity record(s) attached to the Email Message record &lt;br&gt;Use for all Entity record types (for example, customer, contact)</td>
<td>2015.2</td>
</tr>
<tr>
<td>customRecord</td>
<td>Object</td>
<td>optional</td>
<td>The custom record(s) attached to the Email Message record &lt;br&gt;For custom records you must specify both the record ID and the record type ID.</td>
<td>2015.2</td>
</tr>
<tr>
<td>customRecord.id</td>
<td>number</td>
<td>optional</td>
<td>The instance ID for the custom record attached to the Email Message record</td>
<td>2015.2</td>
</tr>
<tr>
<td>customRecord.RecordType</td>
<td>string</td>
<td>optional</td>
<td>The custom record type attached to the Message record</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/email Module Script Sample](#).

```javascript
//Add additional code
...
var recipientEmails = ['msample@netsuite.com',
  'jdoe@netsuite.com',
  'awolfe@netsuite.com,
  'htest@netsuite.com'];
email.sendBulk({
  author: -5,
  recipients: recipientEmails,
  subject: 'Order Status',
  body: 'Your order has been completed.',
  replyTo: 'accounts@netsuite.com'
});
...
//Add additional code
```

**email.sendBulk.promise(options)**

**Method Description**

This method is used to send bulk email asynchronously when a bounceback notification is not required.

**Note:** For information about the parameters and errors thrown for this method, see `email.sendBulk(options)`. For additional information about promises, see [Promise Object](#).

**Returns**

void
email.sendCampaignEvent(options)

Method Description
Method used to send a single "on-demand" campaign email to a specified recipient and return a campaign response ID to track the email.

Email (campaignemail) sublists are not supported. The campaign must use a Lead Nurturing (campaigndrip) sublist.

Note: This API normally uses a bulk email server to send messages. If you need to increase the successful delivery rate of an email, use email.send(options) so that a transactional email server is used.

Returns
A campaign response ID (tracking code) as number
If the email fails to send, the value returned is –1.

Supported Script Types
Client and server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 usage units

Module
N/email Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.campaignEventId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the campaign event.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: The campaign must use a Lead Nurturing (campaigndrip) sublist.</td>
<td></td>
</tr>
<tr>
<td>options.recipientId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the recipient.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### 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></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The recipient's record must contain an email address.

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/email Module Script Sample.

```javascript
//Add additional code
...
email.sendCampaignEvent({
    campaignEventId: -8,
    recipientId: 142,
});
...
//Add additional code
```

#### `email.sendCampaignEvent.promise(options)`

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send a single “on-demand” campaign email asynchronously to a specified recipient and return a campaign response ID to track the email.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>For information about the parameters and errors thrown for this method, see <code>email.sendCampaignEvent(options)</code>. For additional information about promises, see Promise Object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>A campaign response ID (tracking code) as a number. If the email fails to send, the value returned is –1.</th>
</tr>
</thead>
</table>

**Synchronous Version**

`email.sendCampaignEvent(options)`

**Supported Script Types**

Client and server-side scripts  
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

10 usage units

**Module**

N/email Module

**Since**

2015.2

---

### N/encode Module

This module exposes string encoding and decoding functionality. Load the N/encode module when you want to convert a string to another type of encoding.

- N/encode Module Members
- N/encode Module Script Sample
N/encode Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>encode.convert(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Converts a string to another type of encoding and returns the re-encoded string.</td>
</tr>
<tr>
<td>Enum</td>
<td>encode.Encoding</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported encoding specifications.</td>
</tr>
</tbody>
</table>

N/encode Module Script Sample

The following example converts a string to a different encoding.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/encode'],
function(encode) {
  function convertStringToDifferentEncoding() {
    var stringInput = 'TAfA(c)st StriAfA+-g Input';
    var base64EncodedString = encode.convert({
      string: stringInput,
      inputEncoding: encode.Encoding.UTF_8,
      outputEncoding: encode.Encoding.BASE_64
    });
    var hexEncodedString = encode.convert({
      string: stringInput,
      inputEncoding: encode.Encoding.UTF_8,
      outputEncoding: encode.Encoding.HEX
    });
    convertStringToDifferentEncoding();
  }
  convertStringToDifferentEncoding();
});
```

encode.convert(options)

**Method Description**

Converts a string to another type of encoding.

**Returns**

The re-encoded string

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.
N/encode Module

Governance
None

Module
N/encode Module

Since
2015.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.string</td>
<td>string</td>
<td>required</td>
<td>The string to encode.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>string</td>
<td>required</td>
<td>The encoding used on the input string. The default value is UTF_8. Use the encode.Encoding to set the value.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.outputEncoding</td>
<td>string</td>
<td>required</td>
<td>The encoding to apply to the output string. The default value is UTF_8. Use the encode.Encoding to set the value.</td>
<td>2015.1</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/encode Module Script Sample.

```javascript
//Add additional code
...
var hexEncodedString = encode.convert({
  string: stringInput,
  inputEncoding: encode.Encoding.UTF_8,
  outputEncoding: encode.Encoding.HEX
});
...
//Add additional code
```

effect.Encoding

| Enum Description | Holds the string values for the supported character set encoding. This enum is used to set the value of inputEncoding and outputEncoding parameters that are members of the N/crypto Module or N/encode Module. |

Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>
N/encode Module

Since 2015.1

Values

- UTF_8
- BASE_16
- BASE_32
- BASE_64
- BASE_64_URL_SAFE
- HEX

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see **N/encode Module Script Sample**.

```javascript
//Add additional code
...
var reencoded = encode.convert({
    string: LOREM_IPS,
    inputEncoding: encode.Encoding.BASE_64,
    outputEncoding: encode.Encoding.UTF_8
});
...  
//Add additional code
```

N/error Module

Load the error module when you want to create your own custom SuiteScript errors. Use these custom errors in try-catch statements to abort script execution.

- N/error Module Members
- SuiteScriptError Object Members
- UserEventError Object Members
- N/error Module Script Samples

N/error Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>error.SuiteScriptError</td>
<td>Object</td>
<td>Server-side scripts that are not user event scripts</td>
<td>Encapsulates a SuiteScript error thrown by any script type that is not a user event script.</td>
</tr>
<tr>
<td></td>
<td>error.UserEventError</td>
<td>Object</td>
<td>User event scripts</td>
<td>Encapsulates a SuiteScript error thrown by a user event script.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

**SuiteScriptError Object Members**

The following members are called on error.SuiteScriptError.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>SuiteScriptError.name</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>User-defined error code.</td>
</tr>
<tr>
<td></td>
<td>SuiteScriptError.message</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>Text that displays on the SuiteScript Execution Log, in the Details column.</td>
</tr>
<tr>
<td></td>
<td>SuiteScriptError.id</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>Error ID that is automatically generated when a new error is created.</td>
</tr>
<tr>
<td></td>
<td>SuiteScriptError.stack</td>
<td>Array of strings (read-only)</td>
<td>Server-side scripts</td>
<td>A list of method calls that the script is executing when the error is thrown.</td>
</tr>
</tbody>
</table>

**UserEventError Object Members**

The following members are called on error.UserEventError.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>UserEventError.name</td>
<td>string (read-only)</td>
<td>User event scripts</td>
<td>User-defined error code.</td>
</tr>
<tr>
<td></td>
<td>UserEventError.message</td>
<td>string (read-only)</td>
<td>User event scripts</td>
<td>Text that displays on the SuiteScript Execution Log, in the Details column.</td>
</tr>
<tr>
<td></td>
<td>UserEventError.eventType</td>
<td>string (read-only)</td>
<td>User event scripts</td>
<td>User event type (beforeLoad, beforeSubmit, afterSubmit)</td>
</tr>
<tr>
<td></td>
<td>UserEventError.id</td>
<td>string (read-only)</td>
<td>User event scripts</td>
<td>Error ID that is automatically generated when a new error is created.</td>
</tr>
<tr>
<td></td>
<td>UserEventError.recordId</td>
<td>string (read-only)</td>
<td>User event scripts</td>
<td>Internal ID of the submitted record that triggered the script. This property only holds a value when the error is thrown by an afterSubmit user event script.</td>
</tr>
<tr>
<td></td>
<td>UserEventError.stack</td>
<td>Array of strings (read-only)</td>
<td>User event scripts</td>
<td>A list of method calls that the script is executing when the error is thrown.</td>
</tr>
</tbody>
</table>
N/error Module Script Samples

The following example creates an error.

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

require(['N/error'],
    function(error) {

        function createError() {
            var errorObj = error.create({
                name: 'MY_CODE',
                message: 'my error details',
                notifyOff: true
            });
        }
        createError();
    });
```

The following example creates an error if the variable `somevariable` is false. In the `createError()` function's condition is met, the error is logged and then thrown — the script's execution results in an error being thrown.

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

require(['N/error'],
    function(error) {

        function showError() {
            var somevariable = false;

            if (!somevariable) {
                var errObj = error.create({name : error.Type.WRONG_PARAMETER_TYPE, message : 'Wrong parameter type selected.', notifyOff: false});
                log.error('Error: ' + errObj.name , errObj.message);
                throw errObj;
            }
        }
        showError();
    });
```

### error.SuiteScriptError

<table>
<thead>
<tr>
<th><strong>Object Description</strong></th>
<th>Encapsulates a SuiteScript error for any script type that is not a user event script. Use this object in a try-catch statement to abort script execution.</th>
</tr>
</thead>
</table>
Create a new custom error (error.SuiteScriptError) with the error.create(options) method.

The error.create(options) method returns error.SuiteScriptError when it is called in any server-side script that is not a user event script.

Note: When error.create(options) is called in a user event script, it returns error.UserEventError.

For a complete list of this object's methods and properties, see SuiteScriptError Object Members.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>All server-side scripts that are not user event scripts.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

**Module**  
N/error Module  
**Since**  
2015.2

**Syntax**

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
...
//Add additional code
```

**SuiteScriptError.id**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Error ID that is automatically generated when a new error is created. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All server-side scripts that are not user event scripts. For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

**Module**  
N/error Module  
**Since**  
2015.2

**Syntax**

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
```
... var errorObj = error.create(
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
));
log.debug("Error ID: " + errorObj.id);
...
//Add additional code

SuiteScriptError.message

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Text that displays on the SuiteScript Execution Log, in the Details column. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All server-side scripts that are not user event scripts. For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/error Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Message: " + errorObj.message);
...
//Add additional code
```

SuiteScriptError.name

<table>
<thead>
<tr>
<th>Property Description</th>
<th>A user-defined name (error code). This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All server-side scripts that are not user event scripts. For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>
Module  N/error Module
Since  2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create({
  name: 'MY_CODE',
  message: 'my error details',
  notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
//Add additional code
```

**SuiteScriptError.stack**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>A list of method calls that the script is executing when the error is thrown. The most recently executed method is listed at the top. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Array of strings</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All server-side scripts that are not user event scripts. For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/error Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create({
  name: 'MY_CODE',
  message: 'my error details',
  notifyOff: false
});
log.debug("Error Stack: " + errorObj.stack);
...
//Add additional code
```
error.UserEventError

Object Description

Encapsulates a SuiteScript error for user event scripts. Use this object in a try-catch statement to abort script execution. Create a new custom error (error.UserEventError) with the error.create(options) method. The error.create(options) method returns error.UserEventError when it is called in a user event script.

Note: When error.create(options) is called in a server-side script that is not a user event script, it returns error.SuiteScriptError.

For a complete list of this object's methods and properties, see UserEventError Object Members.

Supported Script Types

User event scripts
For more information, see the help topic SuiteScript 2.0 User Event Script Type.

Module

N/error Module

Since

2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
...
//Add additional code
```

UserEventError.eventType

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The user event type. Holds one of the following values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ beforeLoad</td>
<td></td>
</tr>
<tr>
<td>▪ beforeSubmit</td>
<td></td>
</tr>
<tr>
<td>▪ afterSubmit</td>
<td></td>
</tr>
</tbody>
</table>

This property is read-only.

Type

string

Supported Script Types

User event scripts
For more information, see the help topic SuiteScript 2.0 User Event Script Type.

Module

N/error Module
N/error Module

Since 2015.2

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("User Event Type: " + errorObj.eventType);
...
//Add additional code
```

**UserEventError.stack**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>A list of method calls that the script is executing when the error is thrown. The most recently executed method is listed at the top. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Array of strings</td>
</tr>
</tbody>
</table>
| Supported Script Types | User event scripts  
For more information, see the help topic SuiteScript 2.0 User Event Script Type. |
| Module               | N/error Module                                                                                   |
| Since                | 2015.2                                                                                           |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Stack: " + errorObj.stack);
...
//Add additional code
```

**UserEventError.id**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Error ID that is automatically generated when a new error is created.</th>
</tr>
</thead>
</table>
UserEventError.message

- **Property Description**: Text that displays on the SuiteScript Execution Log, in the Details column. This property is read-only.
- **Type**: string
- **Supported Script Types**: User event scripts
- **Module**: N/error Module
- **Since**: 2015.2

### Syntax

**Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/error Module Script Samples](#).

```javascript
//Add additional code
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error ID: " + errorObj.id);
...
//Add additional code
```

```javascript
//Add additional code
...
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Error Message: " + errorObj.message);
...
```
### UserEventError.name

<table>
<thead>
<tr>
<th>Property Description</th>
<th>A user-defined name (error code). This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>
| Supported Script Types | User event scripts  
For more information, see the help topic SuiteScript 2.0 User Event Script Type. |
| Module               | N/error Module                                                   |
| Since                | 2015.2                                                           |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
...
var errorObj = error.create(
  name: 'MY_CODE',
  message: 'my error details',
  notifyOff: false
);
log.debug("Error Code: " + errorObj.name);
...
//Add additional code
```

### UserEventError.recordId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The internal ID of the submitted record that triggered the script. This property only holds a value when the error is thrown by an afterSubmit user event script. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>
| Supported Script Types | User event scripts  
For more information, see the help topic SuiteScript 2.0 User Event Script Type. |
| Module               | N/error Module                                                   |
| Since                | 2015.2                                                           |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/error Module Script Samples.

```javascript
//Add additional code
```
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
    notifyOff: false
});
log.debug("Submitted Record ID: " + errorObj.recordId);

// Add additional code

error.create(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to create a new error.SuiteScriptError or error.UserEventError object. Use this custom error in a try-catch statement to abort script execution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>■ An error.UserEventError object if the script throwing the error is a user event script.</td>
</tr>
<tr>
<td></td>
<td>■ An error.SuiteScriptError object if the script throwing the error is any other server-side script.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/error Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object. The table below describes the name:value pairs that make up the object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>required</td>
<td>■ A user-defined name (error code).</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ Sets the value for the property SuiteScriptError.name or UserEventError.name.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2015.2</td>
</tr>
<tr>
<td>options.message</td>
<td>string</td>
<td>required</td>
<td>■ The error message displayed. This value displays on the Execution Log in the Details column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ The default value is null.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ Sets the value for the property SuiteScriptError.message or UserEventError.message.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

// test
var errorObj = error.create({
    name: 'MY_CODE',
    message: 'my error details',
});
Parameter | Type | Required / Optional | Description
--- | --- | --- | ---
options.notifyOff | boolean | optional | Sets whether email notification is suppressed.
The default value is false.
If set to false, when this error is thrown, the system emails the users identified on the applicable script record's Unhandled Errors subtab. For additional information on the Unhandled Errors subtab, see the help topic Creating a Script Record.

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>MISS_MANDATORY_PARAMETER</td>
<td>A required argument is missing</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var errorObj = error.create({
  name: 'MY_CODE',
  message: 'my error details',
  notifyOff: false
});
log.debug("Error Code: " + errorObj.name);
...
//Add additional code
```

N/file Module

Load the file module when you want to work with files within NetSuite. You can use this module to upload files to the NetSuite file cabinet. You can also use this module to send files as attachments without uploading them to the file cabinet.

A `file.Reader` object, which is returned by `File.getReader()`, can be used for special read operations. Use `File.getSegments(options)` to retrieve iterator of custom segments of file.

Methods that load content in memory, such as `File.getContents()`, have a 10 MB size limit. This limit does not apply when content is streamed, such as when `File.save()` is called.

- N/file Module Members
- File Object Members
- N/file Module Script Sample
## N/file Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>file.Reader</td>
<td>object</td>
<td>Server-side scripts</td>
<td>Encapsulates a reader that you can use to perform special read operations</td>
</tr>
<tr>
<td></td>
<td>file.delete(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Deletes an existing file.File from the NetSuite file cabinet.</td>
</tr>
<tr>
<td></td>
<td>file.Type</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Sets the value of the File.fileType property.</td>
</tr>
</tbody>
</table>

## File Object Members

The following members are called on file.File.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>File.getContents()</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns the content of a file in string format.</td>
</tr>
<tr>
<td></td>
<td>File.lines.iterator()</td>
<td>boolean true</td>
<td>false</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>File.resetStream()</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Resets the file stream to its previous state.</td>
</tr>
<tr>
<td></td>
<td>File.save()</td>
<td>number</td>
<td>Server-side scripts</td>
<td>Saves a new or updated file to the file cabinet.</td>
</tr>
<tr>
<td></td>
<td>File.getReader()</td>
<td>object</td>
<td>Server-side scripts</td>
<td>Returns reader object for read operations.</td>
</tr>
<tr>
<td></td>
<td>File.getSegments(Object)</td>
<td>object</td>
<td>Server-side scripts</td>
<td>Returns iterator of segments that are delimited by the specified separator.</td>
</tr>
<tr>
<td>Property</td>
<td>File.description</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Description of a file.</td>
</tr>
<tr>
<td></td>
<td>File.encoding</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Character encoding on a file.</td>
</tr>
<tr>
<td></td>
<td>File.fileType</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>File type of a file.</td>
</tr>
</tbody>
</table>
## N/file Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>folder</td>
<td>number</td>
<td>Server-side scripts</td>
<td>Internal ID of the folder that houses a file within the NetSuite file cabinet.</td>
</tr>
<tr>
<td>File</td>
<td>id</td>
<td>number (read-only)</td>
<td>Server-side scripts</td>
<td>Internal ID of a file in the NetSuite file cabinet.</td>
</tr>
<tr>
<td>File</td>
<td>isInactive</td>
<td>boolean true</td>
<td>false</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>File</td>
<td>isOnline</td>
<td>boolean true</td>
<td>false</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>File</td>
<td>isText</td>
<td>boolean (read-only)</td>
<td>Server-side scripts</td>
<td>Indicates whether a file type is text-based.</td>
</tr>
<tr>
<td>File</td>
<td>name</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Name of a file.</td>
</tr>
<tr>
<td>File</td>
<td>path</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>Relative path to a file in the NetSuite file cabinet.</td>
</tr>
<tr>
<td>File</td>
<td>size</td>
<td>number (read-only)</td>
<td>Server-side scripts</td>
<td>Size of a file in bytes.</td>
</tr>
<tr>
<td>File</td>
<td>url</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>URL of a file.</td>
</tr>
</tbody>
</table>

### Reader Object Members

The following members are called on `file.Reader`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Reader.prototype.readUntil</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns string from current position to the next occurrence of <code>options.tag</code></td>
</tr>
<tr>
<td>Method</td>
<td>Reader.prototype.readChars</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns the next <code>options.number</code> characters from the current position.</td>
</tr>
</tbody>
</table>

### N/file Module Script Sample

**Note:** These sample scripts use the `require` function so that you can copy into the debugger and test. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

**Example 1**

The following example creates and saves a file to the file cabinet.
Example 2

The following sample creates and saves a file to the file cabinet. It also sets the values of File.isOnline and the File.folder properties.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/file'],
function(file) {
    function createAndSaveFile() {
        var fileObj = file.create({
            name: 'test.txt',
            fileType: file.Type.PLAINTEXT,
            contents: 'Hello World
Hello World'
        });
        fileObj.folder = -15;
        var id = fileObj.save();
        fileObj = file.load({
            id: id
        });
        createAndSaveFile();
    }
    createAndSaveFile();
});
```

Warning: In this example, the folder ID value is hard-coded. For the sample code to run in the SuiteScript Debugger, you must replace this hard-coded value with a valid folder ID from your account.
Example 3

```javascript
require(["N/file", "N/error", "N/log"],
    function (file, error, log)
{

    // In this sample we will compute the total for the
    // second column value in a csv file.
    //
    // date,amount
    // 10/21/14,200.0
    // 10/21/15,210.2
    // 10/21/16,250.3
    //
    // Create the CSV file
    var csvFile = file.create({
        name: 'data.csv',
        contents: 'date,amount\n',
        folder: 39,
        fileType: 'CSV'
    });
    csvFile.appendLine({
        value: '10/21/14,200.0'
    });
    csvFile.appendLine({
        value: '10/21/15,210.2'
    });
    csvFile.appendLine({
        value: '10/21/16,250.3'
    });
    var csvFileId = csvFile.save();

    // This variable will store the total.
    var total = 0.0;

    // Load the file and
    // process all the lines
    var invoiceFile = file.load({
        id: csvFileId
    });
    var iterator = invoiceFile.lines.iterator();

    //Skip the first line (CSV header)
    iterator.each(function () {return false;});
    iterator.each(function (line)
    {
        // This function updates the total by
        // adding the amount on each line to it
        var lineValues = line.value.split(',');
        var lineAmount = parseFloat(lineValues[1]);
    });
```

if (!lineAmount)
    throw error.create({
        name: 'INVALID_INVOICE_FILE',
        message: 'Invoice file contained non-numeric value for total: ' + lineValues[1]
    });

    total += lineAmount;
    return true;
});

// By the time you are here, the total variable is
// set to 660.5
log.debug({
    title: 'total',
    details: total
});

Example 4

/**
 * @NApiVersion 2.0
 * @NScriptType bankStatementParserPlugin
 *
 * A parser plugin that writes chunks of the input file stream to the execution log.
 */
define(['N/file', 'N/log'],
    function(file, log) {
        return {
            parseBankStatement: function(context) {

                var reader = context.input.file.getReader();

                var textUntilFirstComma = reader.readUntil(',');
                var next10Characters = reader.readChars(10);
                var textUntilNextNewLine = reader.readUntil('
');
                var next100Characters = reader.readChars(100);

                log.debug({
                    title: 'STATEMENT TEXT',
                    details: textUntilFirstComma
                });

                log.debug({
                    title: 'STATEMENT TEXT',
                    details: next10Characters
                });

                log.debug({
                    title: 'STATEMENT TEXT',
                    details: textUntilNextNewLine
                });
        }
    })
log.debug({
    title: 'STATEMENT TEXT',
    details: next100Characters
});

Example 5

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType bankStatementParserPlugin
 *
 * A parser plugin that writes segments to the execution log.
 * /
 * define(['N/file', 'N/log'],
 * function(file, log) {
 *     return {
 *         parseBankStatement: function(context) {
 *             var statementFile = context.input.file;
 *             var statementSegmentIterator = statementFile.getSegments({separator: '\|_|/'}).iterator();
 *             statementSegmentIterator.each(function (segment) {
 *                 log.debug({
 *                     title: 'STATEMENT TEXT',
 *                     details: segment.value
 *                 });
 *                 return true;
 *             });
 *         }
 *     }
 * });
 */

file.File

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates a file within NetSuite.</th>
</tr>
</thead>
</table>
| **Note:** This object only encapsulates a file's metadata. Content is only loaded into memory (and returned as a string) when you call the `File.getContents()`. Content from CSV or text files can be accessed line by line using `File.appendLine(options)` or `File.lines.iterator()`.

| **Important:** Binary content must be base64 encoded. |

Create a new `file.File` Object (up to 10MB in size) with the `file.create(options)` method.

After you create a new `file.File`, you can:
upload it to the NetSuite file cabinet with the File.save() method.

attach it to an email or fax without saving it to the file cabinet.

Important: If you want to save the file to the NetSuite file cabinet, you must set a NetSuite file cabinet folder with the File.folder property. You must do this before you call File.save().

Returns reader object File.getReader() and iterator of segments File.getSegments(options).
For a complete list of this object's methods and properties, see File Object Members.

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/file Module

Since
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```
//Add additional code
...
var fileObj = file.create({
    name: 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World
    Hello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...

//Add additional code
```

File.getContents()

Method Description
Method used to return the content of the file.

Important: Content held in memory is limited to 10MB.

Note: You can access CSV or text files (including files over 10MB) using File.appendLine(options) or File.lines.iterator().

Returns
The file content as a string

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/file Module
Since 2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_FILE_CONTENT_SIZE_EXCEEDED</td>
<td>The file content you are attempting to access exceeds the maximum allowed size of 10 MB.</td>
<td>You attempt to return the content of a file larger than 10MB.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.load({
  id: 145
});
if (fileObj.size < 10485760){
  fileObj.getContents();
}
...
//Add additional code
```

File.getReader()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to return the reader object for performing special read operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>file.Reader</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/file Module</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
// Add additional code
...
var reader = context.input.file.getReader();

var textUntilFirstComma = reader.readUntil(',');
var next10Characters = reader.readChars(10);
var textUntilNextNewLine = reader.readUntil('
');
```
```javascript
var next100Characters = reader.readChars(100);
...
// Add additional code
```

### File.getSegments(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method used to return the iterator of segments delimited by a separator. Separator is included in each segment. Empty separator is not allowed.</td>
<td></td>
<td>Server-side scripts</td>
<td>None</td>
<td>N/file Module</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>Iterator</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
</tbody>
</table>

#### Parameters

*Note: The options parameter is a JavaScript object.*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.separator</td>
<td>string</td>
<td>required</td>
<td>The separator to use to divide the segments. For example, if you specify a newline character as the separator, this method returns an iterator where each segment is a single line in the file.</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_SEGMENT_SEPARATOR</td>
<td>Segment separator must not be empty.</td>
<td>The <code>options.separator</code> argument is empty.</td>
</tr>
<tr>
<td>SSS_INVALID_ARG_TYPE</td>
<td>You have entered an invalid type argument: <code>&lt;passed type argument&gt;</code></td>
<td>The <code>options.separator</code> argument is not a string.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td><code>&lt;name of missing parameter&gt;</code></td>
<td>A required argument is not passed.</td>
</tr>
</tbody>
</table>

### Syntax

*Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.*

// Add additional code
var statementFile = context.input.file;

    var statementSegmentIterator = statementFile.getSegments({
        separator: '\|_|/'
    }).iterator();
    statementSegmentIterator.each(function (segment) {

        log.debug({
            title: 'STATEMENT TEXT',
            details: segment.value
        });
        return true;

    });// Add additional code

File.appendLine(options)

Method Description
Method used to insert a line to the end of a file.
This method can be used on text or .csv files.

**Important:** Content held in memory is limited to 10MB. Therefore, each line must be less than 10MB.

<table>
<thead>
<tr>
<th>Returns</th>
<th>file.File Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/file Module</td>
</tr>
<tr>
<td>Since</td>
<td>2017.1</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object. The table below describes the name:value pairs that make up the object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.value</td>
<td>string</td>
<td>required</td>
<td>Object containing a string to insert at the end of the file.</td>
<td>2017.1</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_FILECONTENTSIZEEXCEEDED</td>
<td>You attempt to return the content of a line larger than 10MB.</td>
<td>You attempt to return the content of a line larger than 10MB.</td>
</tr>
</tbody>
</table>
### Error Code

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>YOU_CANNOT_WRITE_TO_A_FILE_AFTER_YOU_BEGAN_READINGFromFile</td>
<td>maximum allowed size of 10 MB.</td>
<td>You call File.appendLine(options) after calling File.lines.iterator(). To avoid receiving the error, call File.resetStream() or save the file.</td>
</tr>
</tbody>
</table>

#### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/file Module Script Sample.`

```javascript
//Add additional code
...
var fileObj = file.load({
  id: 145
});
fileObj.appendLine({
  value: 'hello world'
});
...
//Add additional code
```

### File.lines.iterator()

**Method Description**

Method used to pass the next line as an argument to a developer-defined function. You can call this method multiple times to loop over the file contents as a stream.

Return `false` to stop the loop. Return `true` to continue the loop. By default, false is returned when the end of the file is reached.

This method can be used on text or .csv files.

⚠️ **Important:** Content held in memory is limited to 10MB. Therefore, each line must be less than 10MB.

**Returns**

`Boolean true | false`

**Supported Script Types**

Server-side scripts

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Governance**

None

**Module**

`N/file Module`

**Since**

2017.1

#### 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>lineContext</td>
<td>iterator</td>
<td>required</td>
<td>Iterator which provides the next line of text from the text file to the iterator function.</td>
<td>2017.1</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_FILE_CONTENT_SIZE_EXCEEDED</td>
<td>The content you are attempting to access exceeds the maximum allowed size of 10 MB.</td>
<td>You attempt to return the content of a line larger than 10MB.</td>
</tr>
<tr>
<td>YOU_CANNOT_READ_FROM_FILE_AFTER_YOUBegan_WRITING_IT</td>
<td>You call File.lines.iterator() after calling File.appendLine(options). Call File.resetStream() or save the file.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var iterator = invoiceFile.lines.iterator();

//Skip the first line (CSV header)
iterator.each(function () {return false;});
iterator.each(function (line)
{
    // This function updates the total by adding the amount on each line to it
    var lineValues = line.value.split(',');
    var lineAmount = parseFloat(lineValues[1]);
    if (!lineAmount)
        throw error.create({
            name: 'INVALID_INVOICE_FILE',
            message: 'Invoice file contained non-numeric value for total: ' + lineValues[1]
        });
    total += lineAmount;
    return true;
});
...
//Add additional code
```

File.resetStream()

**Method Description**

Method used to reset the file contents. Serves as an undo action on any unsaved content written with File.appendLine(options) or File.lines.iterator().

Use this method to reset the reading and writing streams that may have been opened by File.appendLine(options) or File.lines.iterator().

The line pointer (or read iterator) is also set to its previous state.

This method can be used on text or .csv files.

⚠️ Important: To use this method, each line must be less than 10MB.

**Returns**

Void
<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For more information, see the help topic SuiteScript 2.0 Script Types.</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/file Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2017.1</th>
</tr>
</thead>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var afile = file.create({
    name: 'tmp3.txt',
    fileTpe: 'PLAINTEXT',
    contents: 'one line'
});
afile.appendLine({
    value: 'line two'
});
afile.resetStream();
afile.lines(function f(){
    ...
//Add additional code
```

### File.save()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ Upload a new file to the NetSuite file cabinet.</td>
<td></td>
</tr>
<tr>
<td>▪ Save an updated file to the NetSuite file cabinet.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The File.save() method streams files of any size, provided that the file to save or upload meets file cabinet limits.

**Important:** If you want to save the file to the NetSuite file cabinet, you must set a NetSuite file cabinet folder with the File.folder property. You must do this before you call File.save().

<table>
<thead>
<tr>
<th>Returns</th>
<th>The internal ID of the file as a number.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>For more information, see the help topic SuiteScript 2.0 Script Types.</strong></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>20 usage units</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/file Module</th>
</tr>
</thead>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_KEY_OR_REF</td>
<td>Invalid folder reference key &lt;passed folder ID&gt;.</td>
<td>The File.folder property is set to an invalid folder ID.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Please enter value(s) for: Folder</td>
<td>The File.folder property is not set before save() is called.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.create({
    name : 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World
Hello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...
//Add additional code
```

File.description

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The description of a file. In the UI, the value of description displays in the Description field on the file record.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/file Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
```

SuiteScript 2.0 API Reference
File.encoding

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
<th>Type</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The character encoding on a file. Value is set with the file.Encoding enum.</td>
<td>string</td>
<td>Server-side scripts</td>
</tr>
</tbody>
</table>

*Supported Script Types*

- Server-side scripts
- For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/file Module

**Since**
2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.create({
    name : 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World
Hello World'
});
fileObj.encoding = file.Encoding.MAC_ROMAN;
fileObj.folder = 30;
var fileId = fileObj.save();
...
//Add additional code
```

File.fileType

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
<th>Type</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The file type of a file. This property is read-only. You must set the file type by passing in a file.Type enum value to file.create(options).</td>
<td>enum</td>
<td>Server-side scripts</td>
</tr>
</tbody>
</table>

*Supported Script Types*

- Server-side scripts
- For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/file Module

**Since**
2015.2
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property after it is set with <code>file.create(options)</code></td>
<td></td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/file Module Script Sample`.

```
//Add additional code
...
var fileObj = file.load({
  id: 145
});
log.debug({
  details: 'File Type: ' + fileObj.fileType
});
...
//Add additional code
```

File.folder

**Property Description**
The internal ID of a file's folder within the NetSuite file cabinet.

Before you upload a file to the NetSuite file cabinet with `file.save()`, you must set its file cabinet folder with the `folder` property.

**Type**
number | string

**Supported Script Types**
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/file Module

**Since**
2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/file Module Script Sample`.

```
//Add additional code
...
var fileObj = file.create({
  name: 'test.txt',
  fileType: file.Type.PLAINTEXT,
  contents: 'Hello World
Hello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
```
File.id

Property Description  The internal ID of the file within the NetSuite file cabinet. This value is automatically generated by NetSuite. This property is read-only.

Type  number

Supported Script Types  Server-side scripts

Module  N/file Module

Since  2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
log.debug({
    details: "File ID: ' + fileObj.id
});
...
//Add additional code
```

File.isInactive

Property Description  The inactive status of a file. If set to true, the file is inactive. The default value is false. When a file is inactive, it does not display in the UI unless you select Show Inactives on the File Cabinet page.

Type  boolean true | false

Supported Script Types  Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.
### File.isOnline

**Property Description**: The Available without Login status of a file. If set to `true`, users can download the file outside of a current NetSuite login session.

The default value is `false`.

**Important**: This property holds the value of the Available without Login setting found on the file record. It does not reflect the value of the Available Without Login setting found on the Suitelet script deployment record.

The Available without Login setting is primarily used for SuiteCommerce websites. When this setting is enabled, websites can access media files in the NetSuite file cabinet without a current NetSuite login session.

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean <code>true</code></th>
<th><code>false</code></th>
</tr>
</thead>
</table>

**Supported Script Types**: Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**: N/file Module

**Since**: 2015.2

---

#### Syntax

**Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/file Module Script Sample](#).

```javascript
// Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.name = 'myOldImageFile.jpg';
fileObj.isInactive = true;
var fileId = fileObj.save();
...
// Add additional code
```
File.isText

#### Property Description
Indicates whether a file type is text-based.
This property is read-only.

#### Type
boolean `true` | `false`

#### Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

#### Module
N/file Module

#### Since
2015.2

---

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

---

### Syntax

```
//Add additional code
...
var fileObj = file.load({
    id: 145
});
if (fileObj.isText === true){
    ...
}...
//Add additional code
```

---

File.name

#### Property Description
The name of a file.

#### Type
string

#### Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

#### Module
N/file Module

#### Since
2015.2
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.name = 'myOldImageFile.jpg';
var fileId = fileObj.save();
...
//Add additional code
```

### File.path

**Property Description**
The relative path to a file in the NetSuite file cabinet.

**Note:** If the folder is not set with the `file.create(options)` method, this property holds the file name until the `File.folder` property is defined.

This property is read-only.

**Type**
string

**Supported Script Types**
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/file Module

**Since**
2015.2

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.load({
    id: 145
});
log.debug({
    details: 'File Path: ' + fileObj.path
});
...
//Add additional code
**File.size**

**Property Description**  
The size of a file in bytes.  
This property is read-only.

**Note:** You can use this value to determine if the file is within size limits for File.getContents(). Size will reflect any lines you have streamed into a file. For example, the original file size plus lines appended.

<table>
<thead>
<tr>
<th>Type</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Module</td>
<td>N/file Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

```
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
log.debug({
    details: 'File Size: ' + fileObj.size
});
...
//Add additional code
```

**File.url**

**Property Description**  
The URL of a file.  
This property is read-only.

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Module</td>
<td>N/file Module</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code ...
var fileObj = file.load({
  id: 'Images/myImagefile.jpg'
});
log.debug({
  details: 'File URL: ' + fileObj.url
});
...
//Add additional code
```

file.create(options)

Method Description

Method used to create a new file in the NetSuite file cabinet.

Important: Content held in memory is limited to 10MB.

Returns

file.File

Supported Script Types

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/file Module

Since

2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>required</td>
<td>The file name and extension.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Sets the value for the File.name property.</td>
<td></td>
</tr>
<tr>
<td>options.fileType</td>
<td>enum</td>
<td>required</td>
<td>The file type.</td>
<td>2015.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>options.contents</td>
<td>string</td>
<td>optional</td>
<td>■ The file content.</td>
<td></td>
</tr>
</tbody>
</table>
|                    |           |                     | ■ File content is lazy loaded; there is no property for it.  
|                    |           |                     | ■ If the file type is binary (for example, PDF), the file content must be base64 encoded.                                                                                                                    |
| options.description| string    | optional            | ■ The file description. In the UI, the value of description displays the Description field on the file record.  
|                    |           |                     | ■ Sets the value for the File.description property.                                                                                                                                                    |
| options.folder     | number    | optional            | ■ The internal ID of the folder within the NetSuite file cabinet. You must set the file cabinet folder before you upload a file to the NetSuite file cabinet with File.save().  
|                    |           |                     | ■ Sets the value for the File.folder property.                                                                                                                                                    |
| options.encoding   | string    | optional            | ■ The character encoding on a file.  
|                    |           |                     | ■ Sets the value for the File.encoding property.                                                                                                                                                    |
|                    |           |                     | ■ Use the file.Encoding enum to set the value.                                                                                                                                                    |
| options.isInactive | boolean   | optional            | ■ The inactive status of a file. If set to true, the file is inactive. The default value is false. When a file is inactive, it does not display in the UI unless you select Show Inactives on the File Cabinet page.  
|                    | false |                     | ■ Sets the value for the File.isInactive property.                                                                                                                                                  |
|                    | true    |                     |                                                                                                                                             |
| options.isOnline   | boolean   | optional            | ■ The Available without Login status of a file. If set to true, users can download the file outside of a current netSuite login session. The default value is false.  
|                    | false |                     | ■ Sets the value for the File.isOnline property.                                                                                                                                                  |
|                    | true    |                     |                                                                                                                                             |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>&lt;name of missing parameter&gt;</td>
<td>A required argument is not passed.</td>
</tr>
<tr>
<td>SSS_INVALID_TYPE_ARG</td>
<td>You have entered an invalid type argument: &lt;passed type argument&gt;</td>
<td>The argument for File.fileType is invalid.</td>
</tr>
<tr>
<td>SSS_FILE_CONTENT_SIZE_EXCEEDED</td>
<td>The file you are trying to create exceeds the maximum allowed file size of 10.0 MB.</td>
<td>You attempt to create a file larger than 10MB.</td>
</tr>
</tbody>
</table>
file.delete(options)

Method Description
Method used to delete an existing file from the NetSuite file cabinet.

Returns
The internal ID of the deleted file

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
20 usage units

Module
N/file Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Internal ID of the file.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To find the internal ID of the file in the UI, click Documents &gt; Files &gt; File Cabinet.</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>&lt;name of missing parameter&gt;</td>
<td>A required argument is not passed.</td>
</tr>
</tbody>
</table>
file.load(options)

**Method Description**
Method used to load an existing file from the NetSuite file cabinet.

**Returns**
An existing file.File.

**Supported Script Types**
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
10 usage units

**Module**
N/file Module

**Since**
2015.2

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSUFFICIENT_PERMISSION</td>
<td>You do not have access to the media item you selected.</td>
<td>Internal ID passed is invalid.</td>
</tr>
<tr>
<td>RCRD_DSNT_EXIST</td>
<td>That record does not exist. path: (path)</td>
<td>Relative file path passed is invalid.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>&lt;name of missing parameter&gt;</td>
<td>A required argument is not passed.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.load({
    id: 'Images/myImageFile.jpg'
});
fileObj.description = 'my test file';
var fileId = fileObj.save();
...
//Add additional code
```

file.Encoding

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This enum is used to set the value of the File.encoding property.</td>
<td></td>
</tr>
</tbody>
</table>

Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

Supported Script Types

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/file Module

Since
2015.2

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Character Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>UTF_8</td>
<td>Unicode</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Character Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>WINDOWS_1252</td>
<td>Western</td>
</tr>
<tr>
<td>ISO_8859_1</td>
<td>Western</td>
</tr>
<tr>
<td>GB18030</td>
<td>Chinese Simplified</td>
</tr>
<tr>
<td>SHIFT_JIS</td>
<td>Japanese</td>
</tr>
<tr>
<td>MAC_ROMAN</td>
<td>Western</td>
</tr>
<tr>
<td>GB2312</td>
<td>Chinese Simplified</td>
</tr>
<tr>
<td>BIG5</td>
<td>Chinese Traditional</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see **N/file Module Script Sample**.

```javascript
//Add additional code
...
var fileObj = file.create({
    name: 'test.txt',
    fileType: file.Type.PLAINTEXT,
    contents: 'Hello World
Hello World'
});
fileObj.encoding = file.Encoding.MAC_ROMAN;
fileObj.folder = 30;
var fileId = fileObj.save();
...
//Add additional code
```

### file.Type

**Enumeration that holds the string values for supported file types. This enum is used to set the value of the File.fileType property.**

Note that the File.fileType property is read only. It's value must be set with file.create(options).

See **N/file Module Script Sample** for an example.

#### Note:

JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/file Module

**Since**

2015.2

---

*Oracle NETSUITE*
Values

- APPCACHE
- AUTOCAD
- BMPIMAGE
- CERTIFICATE
- CONFIG
- CSV
- EXCEL
- FLASH
- FREEMARKER
- GIFIMAGE
- GZIP
- HTMLDOC
- ICON
- JAVASCRIPT
- JPGIMAGE
- JSON
- MESSAGERFC
- MP3
- MPEGMOVIE
- MSPROJECT
- PDF
- PJPIMAGE
- PLAINTEXT
- PNGIMAGE
- POSTSCRIPT
- POWERPOINT
- QUICKTIME
- RTF
- SCSS
- SMS
- STYLESSHEET
- SVG
- TAR
- TIFFIMAGE
- VISIO
- WEBAPPLE
- WEBAPPSCRIPT
- WORD
- XMLDOC
- XSD
- ZIP

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
//Add additional code
...
var fileObj = file.create({
  name: 'test.txt',
  fileType: file.Type.PLAINTEXT,
  contents: 'Hello World\nHello World'
});
fileObj.folder = 30;
var fileId = fileObj.save();
...
//Add additional code
```

file.Reader

**Object Description**

Use for special read operations. Reads from a file until a specified delimiter is reached.

Reads an arbitrary number of characters from a file.

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/file Module

**Methods and Properties**

Reader Object Members
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/file Module Script Sample](#).

```javascript
var reader = context.input.file.getReader();

    var textUntilFirstComma = reader.readUntil(',');
var next10Characters = reader.readChars(10);
var textUntilNextNewLine = reader.readUntil('
');
var next100Characters = reader.readChars(100);
```

### Reader.readUntil(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns string from current position to the next occurrence of <code>options.tag</code>.</td>
<td>Returns the rest of the string if tag is not found.</td>
</tr>
<tr>
<td>Returns null if reading is already finished.</td>
<td>All types of characters are supported. If there's a character that does not exist until the end of the file, the rest of the file is returned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>Server-side scripts</td>
<td>None</td>
<td><a href="#">N/file Module</a></td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.tag</code></td>
<td>string</td>
<td>required</td>
<td>String containing a tag</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_TAG_CANNOT_BE_EMPTY</td>
<td>Tag cannot be empty.</td>
<td>The <code>options.tag</code> argument is empty.</td>
</tr>
<tr>
<td>SSS_INVALID_ARG_TYPE</td>
<td>You have entered an invalid type argument: <code>&lt;passed type argument&gt;</code></td>
<td>The <code>options.tag</code> argument is not a string.</td>
</tr>
</tbody>
</table>
### Error Code

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>&lt;name of missing parameter&gt;</td>
<td>A required argument is not passed.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/file Module Script Sample](#).

```javascript
// Add additional code
...
var reader = context.input.file.getReader();

    var textUntilFirstComma = reader.readUntil(',');
    var next10Characters = reader.readChars(10);
    var textUntilNextNewLine = reader.readUntil('\n');
    var next100Characters = reader.readChars(100);

...
// Add additional code
```

### Reader.readChars(options)

**Method Description**

Returns the next `options.number` characters from the current position.

Returns less than the number if there is not enough characters to read in the file.

Returns null if reading is already finished.

**Returns**

string

**Supported Script Types**

Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

None

**Module**

N/file Module

**Since**

2019.1

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.number</td>
<td>number</td>
<td>required</td>
<td>The number of characters to read.</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_READ_SIZE</td>
<td>Read size must be positive.</td>
<td>The options.number argument is not greater than zero.</td>
</tr>
</tbody>
</table>
### Error Code

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_ARG_TYPE</td>
<td>You have entered an invalid type argument: &lt;passed type argument&gt;</td>
<td>The options.number argument is not a number.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>&lt;name of missing parameter&gt;</td>
<td>A required argument is not passed.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/file Module Script Sample.

```javascript
// Add additional code
...
var reader = context.input.file.getReader();

var textUntilFirstComma = reader.readUntil(',');
var next10Characters = reader.readChars(10);
var textUntilNextNewLine = reader.readUntil('
');
var next100Characters = reader.readChars(100);

...
// Add additional code
```

### N/format Module

Use the format module to parse formatted data into strings and to convert strings into a specified format. The format module formats data according to personal preferences set on the Set Preferences page, accessible from Home > Set Preferences. See the help topic Setting Personal Preferences.

- N/format Module Members
- N/format Module Script Samples

### N/format Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>format.###format###</td>
<td>string</td>
<td>Date</td>
<td>Takes a raw value and returns a formatted value.</td>
</tr>
</tbody>
</table>

**Note:** This method is overloaded when you format a datetime or datetimetz value.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>format.###parse###</td>
<td>Date</td>
<td>string</td>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>
### N/format Module

**Member Type** | **Name** | **Return Type / Value Type** | **Supported Script Types** | **Description**
--- | --- | --- | --- | ---
Enum | format.Type | enum | Client and server-side scripts | Holds the string values for the supported field types. This enum is used to set the value of the `options.type` parameter.

| Enum | format.Timezone | enum | Client and server-side scripts | Holds the string values for supported time zone formats. This enum is used to set the value of the `options.timezone` parameter.

---

### N/format Module Script Samples

For help with writing scripts in SuiteScript 2.0, see the help topics [SuiteScript 2.0 Hello World](#) and [SuiteScript 2.0 Entry Point Script Creation and Deployment](#).

**Example 1**

The following example parses a string (formatted according to the user preference) to a raw Date Object, and then parses it back to the formatted string. This example uses `format.parse(options)` and `format.format(options)`.

**Note:** This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics [SuiteScript 2.0 Script Basics](#) and [SuiteScript 2.0 Script Types](#).

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

require(['N/format'],

function format {
  function parseAndFormatDateString() {
    // Assume Date format is MM/DD/YYYY
    var initialFormattedDateString = "07/28/2015";
    var initialFormattedDateObject = format.parse(
      {value: initialFormattedDateString,
        type: format.Type.DATE}
    );
    var parsedDateStringAsRawDateObject = format.parse(
      {value: initialFormattedDateStringAsRawDateObject,
        type: format.Type.DATE}
    );
    var formattedDateString = format.format(
      {value: parsedDateStringAsRawDateObject,
        type: format.Type.DATE}
    );
    parseAndFormatDateString();
    // "07/28/2015"
  }
});
```
Example 2

The following example parses a string (formatted according to the user preference) to a raw number value, using `format.parse(options)`.

```
Note: This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.
```

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/format'],

function(format){

function parseToValue() {
    // Assume number format is 1.000.000,00 and negative format is -100
    var formattedNum = "-20.000,25"
    return format.parse({value:formattedNum, type: format.Type.FLOAT})
}
var rawNum = parseToValue(); // -20000.25 -- a number
});
```

Example 3

The following example formats a raw number value (formatted according to the user preference) to a string, using `format.format(options)`.

```
Note: This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.
```

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/format'],

function(format){

function formatToString() {
    // Assume number format is 1.000.000,00 and negative format is (100)
    var rawNum2 = -44444.44
    return format.format({value:rawNum2, type: format.Type.FLOAT})
}
var formattedNum2 = formatToString(); // "44.444,44" -- a string
});
```

Example 4

The following example formats the time of day to a string, using `format.format(options)`.

```
Note: This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.
```

```javascript
/**
 * @NApiVersion 2.x
 */
```
require(['N/format'],
    function(format){
        function formatTimeOfDay() {
            // Assume the time format is hh:mm (24 hours)
            var now = new Date(); // Say it's 7:01PM right now.
            return format.format({value: now, type: format.Type.TIMEOFDAY})
        }
        var formattedTime = formatTimeOfDay(); // "19:01" -- a string
    });

format.format(options)

**Method Description**
Formats a value from the raw value to its appropriate preference format.

**Note:** This method is overloaded when you format a `datetime` or `datetimetz` value.

**Returns**
The formatted value as a string.
If a `datetime` or `datetimetz` value is specified, the Date Object is returned in the user's local app time zone.

**Note:** If an invalid value is given, the original value passed to `options.value` is returned.

**Note:** For client side scripts, the string returned is based on the user's system time. For server-side scripts, the string returned is based on the system time of the server your NetSuite system is running on.

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**
None

**Module**
`N/format Module`

**Since**
2015.2

### Parameters
This method is overloaded when you format a `datetime` or `datetimetz` value.

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.value</td>
<td>Date</td>
<td>string</td>
<td>required</td>
<td>The input data to format.</td>
</tr>
<tr>
<td></td>
<td>number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The field type (for example, <code>DATE</code>, <code>CURRENCY</code>, <code>INTEGER</code>). Set using the <code>format.Type</code> enum.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

SuiteScript 2.0 API Reference
The table below applies to datetime and datetimetz values only.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.value</td>
<td>Date</td>
<td>required</td>
<td>The Date Object being converted into a string</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The field type (either DATETIME or DATETIMETZ). Set using the format.Type enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.timezone</td>
<td>enum</td>
<td>number</td>
<td>optional</td>
<td>The time zone specified for the returned string. Set using the format.Timezone enum or key. If a time zone is not specified, the time zone is set based on user preference. If the time zone is invalid, the time zone is set to GMT.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/format Module Script Samples.

```javascript
// Add additional code
...
function(format){
    function formatToString() {
        // Assume number format is 1,000,000,00 and negative format is (100)
        var rawNum2 = -44444.44
        return format.format({value:rawNum2, type: format.Type.FLOAT})
    }
    var formattedNum2 = formatToString(); // "44,444.44" -- a string
    ...
// Add additional code
```

`format.parse(options)`

**Method Description**

Parses a value from the appropriate preference format to its raw value. The appropriate preference format is the one selected in the Date Format field at Home > Set Preferences.

For a datetime or datetimetz value, use this method to convert a Date Object into a string based on the specified timezone.

**Note:** This method is overloaded when you format a datetime or datetimetz value.

**Returns**

The parsed value as a Date | string | number

Datetime or datetimetz values are returned as a string.
**Note:** If the value given is not valid or parseable, the original value passed to `options.value` is returned.

### Supported Script Types
Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types

### Governance
None

### Module
N/format Module

### Since
2015.2

## Parameters

This method is overloaded when you format a `datetime` or `datetimetz` value.

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.value</code></td>
<td>string</td>
<td>required</td>
<td>The input data to parse.</td>
<td>2015.2</td>
</tr>
<tr>
<td><code>options.type</code></td>
<td>string</td>
<td>required</td>
<td>The field type (for example, <code>DATE</code>, <code>CURRENCY</code>, <code>INTEGER</code>).</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set using the <code>format.Type</code> enum.</td>
<td></td>
</tr>
</tbody>
</table>

The table below applies to `datetime` and `datetimetz` values only.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.value</code></td>
<td>string</td>
<td>required</td>
<td>The string that contains the date and time information in the specified timezone.</td>
<td>2015.2</td>
</tr>
<tr>
<td><code>options.type</code></td>
<td>string</td>
<td>required</td>
<td>The field type (either <code>DATETIME</code> or <code>DATETIMETZ</code>).</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set using the <code>format.Type</code> enum.</td>
<td></td>
</tr>
<tr>
<td><code>options.timezone</code></td>
<td>enum</td>
<td>optional</td>
<td>The time zone represented by the <code>options.value</code> string. Set using the <code>format.Timezone</code> enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If a time zone is not specified, the time zone is based on user preference.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If the time zone is invalid, the time zone is set to GMT.</td>
<td></td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/format Module Script Samples.

```javascript
// Add additional code
...
```
function format(
function parseToValue() {
    // Assume number format is 1,000,000,00 and negative format is -100
    var formattedNum = "-20,000,25"
    return format.parse({value: formattedNum, type: format.Type.FLOAT})
}"
    var rawNum = parseToValue(); // -20000.25 -- a number
...
// Add additional code

format.Type

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Enumeration that holds the string values for the supported field types. This enum is used to set the value of the options.type parameter when calling format.format(options) or format.parse(options).</td>
</tr>
</tbody>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client and server-side scripts</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/format Module</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

**Values**

- ADDRESS
- CCEXPDATE
- CCISSN
- CCISSNFROM
- CHECKBOX
- COLOR
- CURRENCY
- CURRENCY2
- DATE
- DATETIME
- DATETIMETZ
- EMAIL
- EMAILS
- FLOAT
- FULLPHONE
- FUNCTION
- FURIGANA
- IDENTIFIER
- IDENTIFIERAN CASE
- INTEGER
- MMYYDATE
- NONNEGCURRENCY
- NONNEGFLOAT
- PACKAGE
- PERCENT
- PHONE
- POSCURRENCY
- POSFLOAT
- POSINTEGER
- QUOTEDFUNCTION
- RADIO
- RATE
- RATEHIGHPRECISION
- TEXT
- TIME
- TIMEOFDAY
- TIMETRACK
- URL

Be aware of the following:

- The following field types require a value of greater than 0:
- POSCURRENCY
- POSINTEGER
- POSINTEGER

- NONNEGFLOAT requires a value that is greater than or equal to 0
- CURRENCY field type rounds the number based on the user’s currency precision setting and is limited to hundredths / 2 decimals (0.00).
- CURRENCY2 field type formats using a record’s currency precision.
- If the RATE or RATEHIGHPRECISION field contains a % in the user interface, it returns null when you get the value.

**Syntax**

**Note:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/format Module Script Samples](#).

```javascript
// Add additional code
...
function formatTimeOfDay() {
    // Assume the time format is hh:mm (24 hours)
    var now = new Date(); // Say it’s 7:01PM right now.
    var formattedTime = format.format({value: now, type: format.Type.TIMEOFDAY})
)}
...
// Add additional code
```

### format.Timezone

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for supported time zone formats. This enum is used to set the value of the <code>options.timezone</code> parameter when calling <code>format.format(options)</code> or <code>format.parse(options)</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a></td>
</tr>
</tbody>
</table>

**Module**

N/format Module

**Since**

2015.2

**Values**

This table defines all valid time zone names in Olson Value format and includes daylight savings time rules for each time zone. Olson Values are maintained by the International Assigned Numbers Authority (IANA) in an international standard time zone database. The values that populate the Time Zone dropdown list found at Home > Set Preferences are also based on these values.
When working with alternate time zones in SuiteScript, use these enumeration values. If necessary, you can use the numerical key in place of an Olson Value string. For example, to source a custom timezone dropdown list.

<table>
<thead>
<tr>
<th>Key</th>
<th>Olson Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ETC_GMT_PLUS_12: 'Etc/GMT+12'</td>
<td>(GMT-12:00) International Date Line West</td>
</tr>
<tr>
<td>2</td>
<td>PACIFIC_SAMOA: 'Pacific/Samoa'</td>
<td>(GMT-11:00) Midway Island, Samoa</td>
</tr>
<tr>
<td>3</td>
<td>PACIFIC_HONOLULU: 'Pacific/Honolulu'</td>
<td>(GMT-10:00) Hawaii</td>
</tr>
<tr>
<td>4</td>
<td>AMERICA_ANCHORAGE: 'America/Anchorage'</td>
<td>(GMT-09:00) Alaska</td>
</tr>
<tr>
<td>5</td>
<td>AMERICA_LOS_ANGELES: 'America/Los_Angeles'</td>
<td>(GMT-08:00) Pacific Time (US &amp; Canada)</td>
</tr>
<tr>
<td>6</td>
<td>AMERICA_TIJUANA: 'America/Tijuana'</td>
<td>(GMT-08:00) Tijuana, Baja California</td>
</tr>
<tr>
<td>7</td>
<td>AMERICA_DENVER: 'America/Denver'</td>
<td>(GMT-07:00) Mountain Time (US &amp; Canada)</td>
</tr>
<tr>
<td>8</td>
<td>AMERICA_PHOENIX: 'America/Phoenix'</td>
<td>(GMT-07:00) Arizona</td>
</tr>
<tr>
<td>9</td>
<td>AMERICA_CHIHUAHUA: 'America/Chihuahua'</td>
<td>(GMT-07:00) Chihuahua, La Paz, Mazatlan - New</td>
</tr>
<tr>
<td>10</td>
<td>AMERICA_CHICAGO: 'America/Chicago'</td>
<td>(GMT-06:00) Central Time (US &amp; Canada)</td>
</tr>
<tr>
<td>11</td>
<td>AMERICA_REGINA: 'America/Regina'</td>
<td>(GMT-06:00) Saskatchewan</td>
</tr>
<tr>
<td>12</td>
<td>AMERICA_GUATEMALA: 'America/Guatemala'</td>
<td>(GMT-06:00) Central America</td>
</tr>
<tr>
<td>13</td>
<td>AMERICA_MEXICO_CITY: 'America/Mexico_City'</td>
<td>(GMT-06:00) Guadalajara, Mexico City, Monterrey - Old</td>
</tr>
<tr>
<td>14</td>
<td>AMERICA_NEW_YORK: 'America/New_York'</td>
<td>(GMT-05:00) Eastern Time (US &amp; Canada)</td>
</tr>
<tr>
<td>15</td>
<td>US_EAST_INDIANA: 'US/East-Indiana'</td>
<td>(GMT-05:00) Indiana (East)</td>
</tr>
<tr>
<td>16</td>
<td>AMERICA_BOGOTA: 'America/Bogota'</td>
<td>(GMT-05:00) Bogota, Lima, Quito</td>
</tr>
<tr>
<td>17</td>
<td>AMERICA_CARACAS: 'America/Caracas'</td>
<td>(GMT-04:30) Caracas</td>
</tr>
<tr>
<td>18</td>
<td>AMERICA_HALIFAX: 'America/Halifax'</td>
<td>(GMT-04:00) Atlantic Time (Canada)</td>
</tr>
<tr>
<td>19</td>
<td>AMERICA_LA_PAZ: 'America/La_Paz'</td>
<td>(GMT-04:00) Georgetown, La Paz, San Juan</td>
</tr>
<tr>
<td>20</td>
<td>AMERICA_MANAUS: 'America/Manaus'</td>
<td>(GMT-04:00) Manaus</td>
</tr>
<tr>
<td>21</td>
<td>AMERICA_SANTIAGO: 'America/Santiago'</td>
<td>(GMT-04:00) Santiago</td>
</tr>
<tr>
<td>22</td>
<td>AMERICA_ST_JOHNS: 'America/St_Johns'</td>
<td>(GMT-03:30) Newfoundland</td>
</tr>
<tr>
<td>23</td>
<td>AMERICA_SAO_PAULO: 'America/Sao_Paulo'</td>
<td>(GMT-03:00) Brasilia</td>
</tr>
<tr>
<td>24</td>
<td>AMERICA_BUENOS_AIRES: 'America/Buenos_Aires'</td>
<td>(GMT-03:00) Buenos Aires</td>
</tr>
<tr>
<td>25</td>
<td>ETC_GMT_PLUS_3: 'Etc/GMT+3'</td>
<td>(GMT-03:00) Cayenne</td>
</tr>
<tr>
<td>26</td>
<td>AMERICA_GODTHAB: 'America/Godthab'</td>
<td>(GMT-03:00) Greenland</td>
</tr>
<tr>
<td>27</td>
<td>AMERICA_MONTEVIDEO: 'America/Montevideo'</td>
<td>(GMT-03:00) Montevideo</td>
</tr>
<tr>
<td>28</td>
<td>AMERICA_NORONHA: 'America/Noronha'</td>
<td>(GMT-02:00) Mid-Atlantic</td>
</tr>
<tr>
<td>29</td>
<td>ETC_GMT_PLUS_1: 'Etc/GMT+1'</td>
<td>(GMT-01:00) Cape Verde Is.</td>
</tr>
<tr>
<td>Key</td>
<td>Olson Value</td>
<td>Description</td>
</tr>
<tr>
<td>-----</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>30</td>
<td>ATLANTIC_AZORES: 'Atlantic/Azores'</td>
<td>(GMT-01:00) Azores</td>
</tr>
<tr>
<td>32</td>
<td>GMT: 'GMT'</td>
<td>(GMT) Casablanca</td>
</tr>
<tr>
<td>33</td>
<td>ATLANTIC_REYKJAVIK: 'Atlantic/Reykjavik'</td>
<td>(GMT) Monrovia, Reykjavik</td>
</tr>
<tr>
<td>34</td>
<td>EUROPE_WARSAW: 'Europe/Warsaw'</td>
<td>(GMT+01:00) Sarajevo, Skopje, Warsaw, Zagreb</td>
</tr>
<tr>
<td>35</td>
<td>EUROPE_PARIS: 'Europe/Paris'</td>
<td>(GMT+01:00) Brussels, Copenhagen, Madrid, Paris</td>
</tr>
<tr>
<td>36</td>
<td>ETC_GMT_MINUS_1: 'Etc/GMT-1'</td>
<td>(GMT+01:00) West Central Africa</td>
</tr>
<tr>
<td>37</td>
<td>EUROPE_AMSTERDAM: 'Europe/Amsterdam'</td>
<td>(GMT+01:00) Amsterdam, Berlin, Bern, Rome, Stockholm, Vienna</td>
</tr>
<tr>
<td>38</td>
<td>EUROPE_BUDAPEST: 'Europe/Budapest'</td>
<td>(GMT+01:00) Belgrade, Bratislava, Budapest, Ljubljana, Prague</td>
</tr>
<tr>
<td>39</td>
<td>AFRICA CAIRO: 'Africa/Cairo'</td>
<td>(GMT+02:00) Cairo</td>
</tr>
<tr>
<td>40</td>
<td>EUROPE_ISTANBUL: 'Europe/Istanbul'</td>
<td>(GMT+02:00) Athens, Bucharest, Istanbul</td>
</tr>
<tr>
<td>41</td>
<td>ASIA_JERUSALEM: 'Asia/Jerusalem'</td>
<td>(GMT+02:00) Jerusalem</td>
</tr>
<tr>
<td>42</td>
<td>ASIA_AMMAN: 'Asia/Amman'</td>
<td>(GMT+02:00) Amman</td>
</tr>
<tr>
<td>43</td>
<td>ASIA_BEIRUT: 'Asia/Beirut'</td>
<td>(GMT+02:00) Beirut</td>
</tr>
<tr>
<td>44</td>
<td>AFRICA_JOHANNESBURG: 'Africa/Johannesburg'</td>
<td>(GMT+02:00) Harare, Pretoria</td>
</tr>
<tr>
<td>45</td>
<td>EUROPE_KIEV: 'Europe/Kiev'</td>
<td>(GMT+02:00) Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius</td>
</tr>
<tr>
<td>46</td>
<td>EUROPE_MINSK: 'Europe/Minsk'</td>
<td>(GMT+02:00) Minsk</td>
</tr>
<tr>
<td>47</td>
<td>AFRICA_WINDHOEK: 'Africa/Windhoek'</td>
<td>(GMT+02:00) Windhoek</td>
</tr>
<tr>
<td>48</td>
<td>ASIA_RIYADH: 'Asia/Riyadh'</td>
<td>(GMT+03:00) Kuwait, Riyadh</td>
</tr>
<tr>
<td>49</td>
<td>EUROPE_MOSCOW: 'Europe/Moscow'</td>
<td>(GMT+03:00) Moscow, St. Petersburg, Volgograd</td>
</tr>
<tr>
<td>50</td>
<td>ASIA_BAGHDAD: 'Asia/Baghdad'</td>
<td>(GMT+03:00) Baghdad</td>
</tr>
<tr>
<td>51</td>
<td>AFRICA_NAIROBI: 'Africa/Nairobi'</td>
<td>(GMT+03:00) Nairobi</td>
</tr>
<tr>
<td>52</td>
<td>ASIA_TEHRAN: 'Asia/Tehran'</td>
<td>(GMT+03:30) Tehran</td>
</tr>
<tr>
<td>53</td>
<td>ASIA_MUSCAT: 'Asia/Muscat'</td>
<td>(GMT+04:00) Abu Dhabi, Muscat</td>
</tr>
<tr>
<td>54</td>
<td>ASIA_BAKU: 'Asia/Baku'</td>
<td>(GMT+04:00) Baku</td>
</tr>
<tr>
<td>55</td>
<td>ASIA_YEREVAN: 'Asia/Yerevan'</td>
<td>(GMT+04:00) Caucasus Standard Time</td>
</tr>
<tr>
<td>56</td>
<td>ETC_GMT_MINUS_3: 'Etc/GMT-3'</td>
<td>(GMT+04:00) Tbilisi</td>
</tr>
<tr>
<td>57</td>
<td>ASIA_KABUL: 'Asia/Kabul'</td>
<td>(GMT+04:30) Kabul</td>
</tr>
<tr>
<td>58</td>
<td>ASIA_KARACHI: 'Asia/Karachi'</td>
<td>(GMT+05:00) Islamabad, Karachi</td>
</tr>
<tr>
<td>59</td>
<td>ASIA_YEKATERINBURG: 'Asia/Yekaterinburg'</td>
<td>(GMT+05:00) Ekaterinburg</td>
</tr>
<tr>
<td>Key</td>
<td>Olson Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>----------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>60</td>
<td>ASIA_TASHKENT: 'Asia/Tashkent'</td>
<td>(GMT+05:00) Tashkent</td>
</tr>
<tr>
<td>61</td>
<td>ASIA_CALCUTTA: 'Asia/Calcutta'</td>
<td>(GMT+05:30) Chennai, Kolkata, Mumbai, New Delhi</td>
</tr>
<tr>
<td>62</td>
<td>ASIA_KATMANDU: 'Asia/Katmandu'</td>
<td>(GMT+05:45) Kathmandu</td>
</tr>
<tr>
<td>63</td>
<td>ASIA_ALMATY: 'Asia/Almaty'</td>
<td>(GMT+06:00) Novosibirsk</td>
</tr>
<tr>
<td>64</td>
<td>ASIA_DHAKA: 'Asia/Dhaka'</td>
<td>(GMT+06:00) Astana, Dhaka</td>
</tr>
<tr>
<td>65</td>
<td>ASIA_RANGOON: 'Asia/Rangoon'</td>
<td>(GMT+06:30) Yangon (Rangoon)</td>
</tr>
<tr>
<td>66</td>
<td>ASIA_BANGKOK: 'Asia/Bangkok'</td>
<td>(GMT+07:00) Bangkok, Hanoi, Jakarta</td>
</tr>
<tr>
<td>67</td>
<td>ASIA_KRASNOYARSK: 'Asia/Krasnoyarsk'</td>
<td>(GMT+07:00) Krasnoyarsk</td>
</tr>
<tr>
<td>68</td>
<td>ASIA_HONG_KONG: 'Asia/Hong_Kong'</td>
<td>(GMT+08:00) Beijing, Chongqing, Hong Kong, Urumqi</td>
</tr>
<tr>
<td>69</td>
<td>ASIA_KUALA_LUMPUR: 'Asia/Kuala_Lumpur'</td>
<td>(GMT+08:00) Kuala Lumpur, Singapore</td>
</tr>
<tr>
<td>70</td>
<td>ASIA_TAIPEI: 'Asia/Taipei'</td>
<td>(GMT+08:00) Taipei</td>
</tr>
<tr>
<td>71</td>
<td>AUSTRALIA_PERTH: 'Australia/Perth'</td>
<td>(GMT+08:00) Perth</td>
</tr>
<tr>
<td>72</td>
<td>ASIA_IRKUTSK: 'Asia/Irkutsk'</td>
<td>(GMT+08:00) Irkutsk</td>
</tr>
<tr>
<td>73</td>
<td>ASIA_MANILA: 'Asia/Manila'</td>
<td>(GMT+08:00) Manila</td>
</tr>
<tr>
<td>74</td>
<td>ASIA_SEOUL: 'Asia/Seoul'</td>
<td>(GMT+09:00) Seoul</td>
</tr>
<tr>
<td>75</td>
<td>ASIA_TOKYO: 'Asia/Tokyo'</td>
<td>(GMT+09:00) Osaka, Sapporo, Tokyo</td>
</tr>
<tr>
<td>76</td>
<td>ASIA_YAKUTSK: 'Asia/Yakutsk'</td>
<td>(GMT+09:00) Yakutsk</td>
</tr>
<tr>
<td>77</td>
<td>AUSTRALIA_DARWIN: 'Australia/Darwin'</td>
<td>(GMT+09:30) Darwin</td>
</tr>
<tr>
<td>78</td>
<td>AUSTRALIA_ADELAIDE: 'Australia/Adelaide'</td>
<td>(GMT+09:30) Adelaide</td>
</tr>
<tr>
<td>79</td>
<td>AUSTRALIA_SYDNEY: 'Australia/Sydney'</td>
<td>(GMT+10:00) Canberra, Melbourne, Sydney</td>
</tr>
<tr>
<td>80</td>
<td>AUSTRALIA_BRISBANE: 'Australia/Brisbane'</td>
<td>(GMT+10:00) Brisbane</td>
</tr>
<tr>
<td>81</td>
<td>AUSTRALIA_HOBART: 'Australia/Hobart'</td>
<td>(GMT+10:00) Hobart</td>
</tr>
<tr>
<td>82</td>
<td>PACIFIC_GUAM: 'Pacific/Guam'</td>
<td>(GMT+10:00) Guam, Port Moresby</td>
</tr>
<tr>
<td>83</td>
<td>ASIA_VLADIVOSTOK: 'Asia/Vladivostok'</td>
<td>(GMT+10:00) Vladivostok</td>
</tr>
<tr>
<td>84</td>
<td>ASIA_MAGADAN: 'Asia/Magadan'</td>
<td>(GMT+11:00) Magadan, Solomon Is., New Caledonia</td>
</tr>
<tr>
<td>85</td>
<td>PACIFIC_KWAJALEIN: 'Pacific/Kwajalein'</td>
<td>(GMT+12:00) Fiji, Marshall Is.</td>
</tr>
<tr>
<td>86</td>
<td>PACIFIC_AUCKLAND: 'Pacific/Auckland'</td>
<td>(GMT+12:00) Auckland, Wellington</td>
</tr>
<tr>
<td>87</td>
<td>PACIFIC_TONGATAPU: 'Pacific/Tongatapu'</td>
<td>(GMT+13:00) Nuku'alofa</td>
</tr>
</tbody>
</table>
N/format Module

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/format Module Script Samples.

```javascript
// Add additional code
...
var date = new Date(); //Mon Aug 24 2015 17:27:16 GMT-0700 (Pacific Daylight Time)
  var TOKYO = format.format({
    value: date,
    type: format.Type.DATETIME,
    timezone: format.Timezone.ASIA_TOKYO
  }); //Returns "8/25/2015 9:27:16 am"

  var NEWYORK = format.format({
    value: date,
    type: format.Type.DATETIME,
    timezone: format.Timezone.AMERICA_NEW_YORK
  }); //Returns "8/24/2015 8:27:16 pm"

var dateStr = "03/17/2015 09:00:00 pm"
  var TOKYO_2 = format.parse({
    value: dateStr,
    type: format.Type.DATETIME,
    timezone: format.Timezone.ASIA_TOKYO
  }); //Returns Date object [[ Tue Mar 17 2015 05:00:00 GMT-0700 (PDT) ]]

  var NEWYORK_2 = format.parse({
    value: dateStr,
    type: format.Type.DATETIME,
    timezone: format.Timezone.AMERICA_NEW_YORK
  }); //Returns Date object [[ Tue Mar 17 2015 18:00:00 GMT-0700 (PDT) ]]
...
// Add additional code
```

N/format/i18n Module

The N/format/i18n module allows for formatting of strings in international context.

- N/format/i18n Module Members

N/format/i18n Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>format.spellOut</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Creates a string containing the spelled-out version of the specified number in a specified locale.</td>
</tr>
</tbody>
</table>

N/format/i18n Script Samples

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.
The following script sample spells out the number 12345 as a string in German, “zwölftausenddrei-
hundertfünfundvierzig”.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/format/i18n'],
function(format) {
    var spellOut = format.spellOut({
        number: 12345,
        locale: "DE"
    });
    log.debug(spellOut);
});
```

### format.spellOut(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Methods and Properties</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Spells out positive and negative number as a string in a specific language. For more information, see <a href="#">Codes for the Representation of Names of Languages</a>.</td>
<td>String</td>
<td>Client and server-side scripts, <a href="#">SuiteScript 2.0 Script Types</a>.</td>
<td>None</td>
<td>N/format Module</td>
<td><a href="#">N/format/i18n Module Members</a></td>
<td>2019.1</td>
</tr>
</tbody>
</table>

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.number</td>
<td>number</td>
<td>required</td>
<td>The number to be spelled out in a string.</td>
<td>2019.1</td>
</tr>
<tr>
<td>options.locale</td>
<td>string</td>
<td>required</td>
<td>The language code that specifies the string's language. ISO 639-1 alpha-2 language codes are supported. The language specified in this parameter is not related to the language specified for a NetSuite account. You can specify any language for this parameter; you do not have to specify a NetSuite supported language. For more information, see <a href="#">Codes for the Representation of Names of Languages</a>.</td>
<td>2019.1</td>
</tr>
</tbody>
</table>
N/http Module

Use the http module to make HTTP calls from server-side or client-side scripts. On the client-side, this module also provides the ability to make cross-domain HTTP requests using NetSuite servers as a proxies.

All HTTP content types are supported.

Note: The http module does not accept the HTTPS protocol. Use the N/https Module for that purpose.

- N/http Module Members
- ClientResponse Object Members
- ServerRequest Object Members
- ServerResponse Object Members
- N/http Module Script Sample

General HTTP Header Blacklist

Be aware that certain headers cannot be set manually when using http module methods. If a script attempts to set values for any of the following headers, the values are discarded. These headers are described in the following table.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection</td>
<td></td>
</tr>
<tr>
<td>Content-Length</td>
<td></td>
</tr>
<tr>
<td>Host</td>
<td></td>
</tr>
<tr>
<td>Trailer</td>
<td></td>
</tr>
<tr>
<td>Transfer-Encoding</td>
<td></td>
</tr>
<tr>
<td>Upgrade</td>
<td></td>
</tr>
<tr>
<td>Via</td>
<td></td>
</tr>
</tbody>
</table>

Suitelet Response HTTP Header Blacklist

In addition to the headers described in General HTTP Header Blacklist, certain headers cannot be set manually when interacting with the http.ServerResponse objects sent by Suitelets. If a script attempts to set values for any of these headers, the system throws an SSS_INVALID_HEADER error. These headers are described in the following table.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access-Control-Allow-Origin</td>
<td></td>
</tr>
<tr>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td></td>
</tr>
<tr>
<td>Content-Length</td>
<td></td>
</tr>
<tr>
<td>Content-Location</td>
<td></td>
</tr>
<tr>
<td>Content-MD5</td>
<td></td>
</tr>
<tr>
<td>Content-Range</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Proxy-Authenticate</td>
<td></td>
</tr>
<tr>
<td>Retry-After</td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Trailer</td>
<td></td>
</tr>
<tr>
<td>Via</td>
<td></td>
</tr>
<tr>
<td>Warning</td>
<td></td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td></td>
</tr>
</tbody>
</table>

N/http Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>http.ClientResponse</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>Encapsulates the response to an HTTP client request.</td>
</tr>
<tr>
<td></td>
<td>http.ServerRequest</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>Encapsulates the HTTP request information sent to an HTTP server.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>http.ServerResponse</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>For example, a request received by a Suitelet or RESTlet.</td>
</tr>
<tr>
<td>Enum</td>
<td>http.RedirectType</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported NetSuite resources that you can redirect to. This enum is used to set the value of the <code>type</code> argument for <code>ServerResponse.sendRedirect(options)</code>.</td>
</tr>
<tr>
<td>Enum</td>
<td>http.Method</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported HTTP requests. This enum is used to set the value of <code>http.request(options)</code> and <code>ServerRequest.method</code>.</td>
</tr>
<tr>
<td>Enum</td>
<td>http.CacheDuration</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported cache durations. This enum is used to set the value of the <code>ServerResponse.setCdnCacheable(options)</code> property.</td>
</tr>
</tbody>
</table>

**ClientResponse Object Members**

The following members are called on `http.ClientResponse`. 
<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>ClientResponse.body</td>
<td>read-only string</td>
<td>Server-side scripts</td>
<td>The client response body.</td>
</tr>
<tr>
<td></td>
<td>ClientResponse.code</td>
<td>read-only number</td>
<td>Server-side scripts</td>
<td>The client response code.</td>
</tr>
<tr>
<td></td>
<td>ClientResponse.headers</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>The client response headers.</td>
</tr>
</tbody>
</table>

ServerRequest Object Members

The following members are called on the `http.ServerRequest` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>ServerRequest.getLineCount(options)</td>
<td>number</td>
<td>Server-side scripts</td>
<td>Returns the number of lines in a sublist.</td>
</tr>
<tr>
<td></td>
<td>ServerRequest.getSublistValue(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns the value of a sublist line item.</td>
</tr>
<tr>
<td>Property</td>
<td>ServerRequest.body</td>
<td>read-only string</td>
<td>Server-side scripts</td>
<td>The server request body.</td>
</tr>
<tr>
<td></td>
<td>ServerRequest.files</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>The server request files.</td>
</tr>
<tr>
<td></td>
<td>ServerRequest.headers</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>The server request headers.</td>
</tr>
<tr>
<td></td>
<td>ServerRequest.parameters</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>The server request parameters.</td>
</tr>
<tr>
<td></td>
<td>ServerRequest.url</td>
<td>read-only string</td>
<td>Server-side scripts</td>
<td>The server request URL.</td>
</tr>
</tbody>
</table>

ServerResponse Object Members

The following members are called on the `http.ServerResponse` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>ServerResponse.addHeader(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Adds a header to the response.</td>
</tr>
<tr>
<td></td>
<td>ServerResponse.getHeader(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Returns the value of a response header.</td>
</tr>
<tr>
<td></td>
<td>ServerResponse.renderPdf(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Generates and renders a PDF directly to the response.</td>
</tr>
<tr>
<td></td>
<td>ServerResponse.sendRedirect(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Sets the redirect URL by resolving to a NetSuite resource.</td>
</tr>
</tbody>
</table>
### N/http Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ServerResponse.###setCdnCacheable###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Sets CDN caching for a period of time.</td>
<td></td>
</tr>
<tr>
<td>ServerResponse.###setHeader###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Sets the value of a response header.</td>
<td></td>
</tr>
<tr>
<td>ServerResponse.###write###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Writes information (text/xml/html) to the response.</td>
<td></td>
</tr>
<tr>
<td>ServerResponse.###writeFile###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Writes a file to the response.</td>
<td></td>
</tr>
<tr>
<td>ServerResponse.###writeLine###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Writes line information (text/xml/html) to the response.</td>
<td></td>
</tr>
<tr>
<td>ServerResponse.###writePage###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Generates a page.</td>
<td></td>
</tr>
</tbody>
</table>

| Property | ServerResponse.headers | Object | Server-side scripts | The server response headers. |

### N/http Module Script Sample

**Example 1**

The following example shows an HTTP GET request for a URL.

> Note: This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

```javascript
/** *
 * @NApiVersion 2.x
 */
require(['N/http'], function(http) {
    function sendGetRequest() {
        var response = http.get({
            url: 'http://www.google.com'
        });
        sendGetRequest();
    }
    sendGetRequest();
});
```

**Example 2**

The following example is designed to redirect to new sales order record, and will set entity to 6. (Assuming there is an entity with number 6, if there's not, then entity will remain blank.)
Note: This sample script uses the `define` function. Note that you cannot use On Demand Debugging to step through a `define` function. You must use Deployed Debugging to step through this script.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define([ 'N/record', 'N/http' ],
    function(record, http) {
        function onRequest(context) {
            context.response.sendRedirect({
                type: http.RedirectType.RECORD,
                identifier: record.Type.SALES_ORDER,
                parameters: {
                    entity: 6
                }
            });
        }
        return {
            onRequest: onRequest
        };}
)
```

### http.ClientResponse

**Object Description**

Encapsulates the response to an HTTP client request.

This object is read-only.

For a complete list of this object's properties, see `ClientResponse Object Members`.

**Supported Script Types**

Server-side scripts

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Module**

`N/http Module`

**Since**

2015.2

### Syntax

*Important:* The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/http Module Script Sample`.

```javascript
//Add additional code
...
var clientResponse = http.get({
    url: 'http://www.google.com'
});
...
//Add additional code
```
### ClientResponse.body

**Property Description**

The client response body.

This property is read-only.

**Type**

string

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/http Module

**Parent Object**

http.ClientResponse

**Sibling Object Members**

ClientResponse Object Members

**Since**

2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
var response = http.get({
  url: 'http://www.google.com'
});
log.debug({
  title: 'Client Response Body',
  details: http.response.body
});
...
//Add additional code
```

### ClientResponse.code

**Property Description**

The client response code.

This property is read-only.

**Type**

number

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/http Module
**ClientResponse.headers**

The response header or headers. This property is read-only.

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Type</th>
<th>Supported Script Types</th>
<th>Module</th>
<th>Parent Object</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>object</td>
<td>Server-side scripts</td>
<td>N/http Module</td>
<td>http.ClientResponse</td>
<td>ClientResponse Object Members</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/http Module Script Sample](#).

```javascript
//Add additional code
...
var response = http.get({
    url: 'http://www.google.com'
});
log.debug({
    title: 'Client Response Header',
    details: http.response.headers
});
...

//Add additional code
```

---

**http.ServerRequest**

**Object Description**

Encapsulates the HTTP request information to an HTTP server. For example, a request received by a Suitelet or RESTlet.

This object is read-only.

For a complete list of this object’s methods and properties, see [ServerRequest Object Members](#).

**Supported Script Types**

Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/http Module

**Since**

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/http Module Script Sample](#).

```javascript
//Add additional code
...
serverRequest.getLineCount({
    group: 'sublistId'
});
...

//Add additional code
```

---

**ServerRequest.getLineCount(options)**

**Method Description**

Method used to return the number of lines in a sublist.

**Returns**

The number of lines in a sublist as a number.
Supported Script Types

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/http Module

Parent Object

http.ServerRequest

Sibling Object Members

ServerRequest Object Members

Since

2015.2

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>options.group</td>
<td>string</td>
<td>required</td>
<td>The sublist internal ID.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
serverRequest.getLineCount({
  group: 'sublistId'
});
...
//Add additional code
```

ServerRequest.getSublistValue(options)

Method Description

Method used to return the value of a sublist line item.

Returns

The value of the sublist line item as a string.

Supported Script Types

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/http Module

Parent Object

http.ServerRequest

Sibling Object Members

ServerRequest Object Members
Since 2015.2

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>options.group</td>
<td>string</td>
<td>required</td>
<td>The sublist internal ID.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.name</td>
<td>string</td>
<td>required</td>
<td>The sublist line item ID (name of the field).</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>string</td>
<td>required</td>
<td>The sublist line number.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** Sublist index starts at 0.

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
serverRequest.getSublistValue({
    group: 'item',
    name: 'amount',
    line: '2'
});
...
//Add additional code
```

ServerRequest.body

**Property Description**
The server request body.

This property is read-only.

**Type**
string

**Supported Script Types**
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/http Module

**Parent Object**
http.ServerRequest

**Sibling Object Members**
ServerRequest Object Members

**Since**
2015.2
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/http Module Script Sample`.

```javascript
//Add additional code
...
log.debug({
  title: 'Server Request Body',
  details: http.request.body
});
...
//Add additional code
```

**ServerRequest.files**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The server request files. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <strong>SuiteScript 2.0 Script Types</strong>.</td>
</tr>
<tr>
<td>Module</td>
<td>N/http Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>http.ServerRequest</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>ServerRequest Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippets show the syntax for this member. They are not functional examples. For a complete script example, see `N/http Module Script Sample`.

```javascript
//Add additional code
...
log.debug({
  title: 'Server Request Files',
```
details: http.request.files

```javascript
var file = request.files[‘file_id’];
```

### ServerRequest.headers

<table>
<thead>
<tr>
<th>Property Description</th>
<th>This object represents a series of key/value pairs. Each pair represents a server request header name and its value. Typically, this object encapsulates two iterations of each header name: one in lower case and another in title case. This behavior is designed so that you can use either lower case or title case when you reference a header. However, the existence of title-case iterations of header names is not guaranteed. For best results, refer to header names using all lower-case letters (and hyphens, when applicable). This property is read-only.</th>
</tr>
</thead>
</table>

**Important:** The server request headers and their values are subject to change. If you use these headers in your scripts, you are responsible for testing them to make sure that they contain the information you need. For example, when making an HTTP call to a Suitelet, some headers might be filtered out. Filtering can occur if the headers affect how NetSuite processes the request internally. These filtered headers are not available to the Suitelet, so you should test to see whether a header was filtered out. If so, use a different header instead.

<table>
<thead>
<tr>
<th>Type</th>
<th>Object</th>
</tr>
</thead>
</table>

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/http Module

**Parent Object**

http.ServerRequest

**Sibling Object Members**

ServerRequest Object Members

**Since**

2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
```

---

**SuiteScript 2.0 API Reference**

---

**Oracle® NetSuite**
ServerRequest.method

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The server request http method. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>http.Method enum</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Module               | N/http Module                                           |
| Parent Object        | http.ServerRequest                                     |
| Sibling Object Members | ServerRequest Object Members |
| Since                | 2015.2                                                 |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

Syntax

```
//Add additional code
...
log.debug({
    title: 'Server Request Method',
    details: http.request.method
});
...
//Add additional code
```

ServerRequest.parameters

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The server request parameters. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Object</td>
</tr>
</tbody>
</table>
Supported Script Types

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/http Module

Parent Object
http.ServerRequest

Sibling Object Members
ServerRequest Object Members

Since
2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
log.debug({
    title: 'Server Request Parameters',
    details: http.request.parameters
});
...
//Add additional code
```

**ServerRequest.url**

**Property Description**
The server request URL.
This property is read-only.

**Type**
string

**Supported Script Types**
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/http Module

Parent Object
http.ServerRequest

Sibling Object Members
ServerRequest Object Members

Since
2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

SuiteScript 2.0 API Reference
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
log.debug({
    title: 'Server Request URL',
    details: http.request.url
});
...
//Add additional code
```

http.ServerResponse

**Object Description**
Encapsulates the response to an incoming http request from an HTTP server. For example, a response from a Suitelet or RESTlet.

For a complete list of this object's methods and properties, see ServerResponse Object Members.

**Supported Script Types**
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/http Module

**Since**
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
//Add additional code
```

ServerResponse.addHeader(options)

**Method Description**
Method used to add a header to the response.

If the same header has already been set, this method adds another line for that header. For example:

```
Vary: 'Accept-Language'
Vary: 'Accept-Encoding'
```
Returns

Returns Void

Supported Script Types

Supported Script Types Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

Governance None

Module

Module N/http Module

Parent Object

Parent Object http.ServerResponse

Sibling Object Members

Sibling Object Members ServerResponse Object Members

Since

Since 2015.2

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>options.name</td>
<td>string</td>
<td>required</td>
<td>The name of the header.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>required</td>
<td>The value used to set the header.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>SSS_INVALID_HEADER</td>
<td>One or more headers are not valid.</td>
<td>The header name or value is invalid.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
serverResponse.addHeader({
  name: 'Accept-Language',
  value: 'en-us',
});
...
//Add additional code
```

ServerResponse.getHeader(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to return the value or values of a response header. If multiple values are assigned to the header name, the values are returned as an Array.</th>
</tr>
</thead>
</table>

| Returns            | string | string[] |
Supported Script Types

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/http Module

Parent Object

http.ServerResponse

Sibling Object Members

ServerResponse Object Members

Since

2015.2

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>options.name</td>
<td>string</td>
<td>required</td>
<td>The name of the header.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: {param name}</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
serverResponse.getHeader({
    name: 'Accept-Language'
});
...
//Add additional code
```

ServerResponse.sendRedirect(options)

**Method Description**

Method used to set the redirect URL by resolving to a NetSuite resource.
For example, you could use this method and your own parameters to make a redirect to a url for associated records, such as a redirect to a new sales order page for a particular entity.

**Returns**

Void

**Supported Script Types**

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The type of resource redirected to. Set this value using the <code>http.RedirectType</code> enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.identifier</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The primary ID for the resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If redirecting to a media item (for example, an image or PDF file), pass in the file id.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If redirecting to a record, pass in the record type using the <code>record.Type</code> enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If redirecting to a RESTlet, pass in the script ID as a number or string.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If redirecting to a tasklink, pass in the task ID. For a list of supported task IDs, see the help topic <a href="https://help.netsuite.com/article/en/SuiteLet/Task-IDs-1184883108">Task IDs</a>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If redirecting to a Suitelet, pass in the script ID.</td>
<td></td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>string</td>
<td>optional</td>
<td>The secondary ID for this resource. If the resource type is a Suitelet or RESTlet, pass in the deployment ID.</td>
</tr>
<tr>
<td>options.editMode</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>Applicable when redirecting to a record resource.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Specifies whether to return a URL for a record in edit mode or view mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default value is <code>false</code> – returns the record in view mode and not edit mode.</td>
<td></td>
</tr>
<tr>
<td>options.parameters</td>
<td>object</td>
<td>optional</td>
<td>Additional URL parameters as name/value pairs.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQU_ARGUMENT</td>
<td>Missing a required argument: {param name}</td>
<td>A required parameter is missing. Note that this error is thrown if the script includes a typo in the <code>options</code> parameter.</td>
<td></td>
</tr>
</tbody>
</table>
Error Code | Message | Thrown If
--- | --- | ---

SSS_INVALID_URL_CATEGORY | The options.type: (type) is not valid. Please use the RedirectType enum for supported types. | The script uses an unrecognizable string value for the options.type parameter. To avoid this error, use the http.RedirectType enum.

SSS_INVALID_TASK_ID | The task ID: (id) is not valid. Please refer to the documentation for a list of supported task IDs. | The type is set to tasklink, and an invalid task ID is input for options.identifier.

SSS_INVALID_RECORD_TYPE | Type argument (type) is not a valid record or is not available in your account. Please see the documentation for a list of supported record types. | The redirect type is set to record, and an invalid record type is input for options.identifier.

SSS_INVALID_SCRIPT_ID_1 | You have provided an invalid script id or internal id: (id) | The type is set to Suitelet or RESTlet, and an invalid script ID or invalid deployment ID is input for options.identifier or options.id.

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```
//Add additional code
...
myServerResponseObj.sendRedirect({
  type: http.RedirectType.RECORD,
  identifier: record.Type.SALES_ORDER,
  parameters: {entity: 8}
});
...
//Add additional code
```

ServerResponse.setHeader(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to set the value of a response header.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/http Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>http.ServerResponse</td>
</tr>
</tbody>
</table>
## 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>options.name</td>
<td>string</td>
<td>required</td>
<td>The name of the header.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>required</td>
<td>The value used to set the header.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>SSS_INVALID_HEADER</td>
<td>One or more headers are not valid.</td>
<td>The header name or value is invalid.</td>
</tr>
</tbody>
</table>

## Syntax

### Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
serviceResponse.setHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
//Add additional code
```

## ServerResponse.renderPdf(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to generate and render a PDF directly to the response.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>10 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/http Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>http.ServerResponse</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>ServerResponse Object Members</td>
</tr>
</tbody>
</table>
Since 2015.2

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>options.xmlString</td>
<td>string</td>
<td>required</td>
<td>Content of the pdf.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code shows the syntax for this member. It is not a functional example. For a complete script example, see [N/http Module Script Sample](#).

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType suitelet
 */
define(['N/xml'], function(xml){
    return {
        onRequest: function(context){
            var xml = '<?xml version="1.0" encoding="UTF-8"?>
            <!DOCTYPE pdf PUBLIC "-//big.faceless.org//report" "report-1.1.dtd">
            <pdf lang="ru-RU" xml:lang="ru-RU">
            <head>
                <link name="russianfont" type="font" subtype="opentype" src="NetSuiteFonts/verdana.ttf" src-bold="NetSuiteFonts/verdanab.ttf" src-italic="NetSuiteFonts/verdanai.ttf" src-bolditalic="NetSuiteFonts/verdanabi.ttf" bytes="2"/>
            </head>
            <body font-family="russianfont" font-size="18">
            Russkii tekst</body>
            </pdf>
            ;
            context.response.renderPdf(xml);
        }
    }
});
```

ServerResponse.setCdnCacheable(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to set CDN caching for a period of time.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
</tbody>
</table>
### Supported Script Types

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
</table>

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

### Governance

None

### Module

N/http Module

### Parent Object

http.ServerResponse

### Sibling Object Members

ServerResponse Object Members

### Since

2015.2

### 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>options.type</td>
<td>enum</td>
<td>required</td>
<td>The value of the caching duration. Set using the http.CacheDuration enum.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

### Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/http Module Script Sample](#).

```javascript
//Add additional code
...
serverResponse.setCdnCacheable({
  type: http.CacheDuration.MAX
});
...
//Add additional code
```

### ServerResponse.write(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to write information to the response.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>This method accepts only strings. To pass in a file, you can use ServerResponse.writeFile(options).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>Void</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
</table>

For more information, see the help topic [SuiteScript 2.0 Script Types](#).
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>options.output</td>
<td>string</td>
<td>required</td>
<td>The output string being written.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>(param name)</td>
<td>The value input for options.output is not a string.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see **N/http Module Script Sample**.

```javascript
//Add additional code
...
serverResponse.write({
    output: 'Hello World'
});
...
//Add additional code
```

ServerResponse.writeFile(options)

Method Description

Method used to write a file to the response.

Returns

Void

Supported Script Types

Server-side scripts

For more information, see the help topic **SuiteScript 2.0 Script Types**.
## 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>options.file</td>
<td>file.File</td>
<td>required</td>
<td>A file.File Object that encapsulates the file to be written.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.isinline</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether the file is inline. If true, the file is inline. The default value is false.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>(param name)</td>
<td>The value input for options.file is not a file.File Object.</td>
</tr>
</tbody>
</table>

## Syntax

```javascript
//Add additional code
...
serverResponse.writeFile({
  file: myFileObj,
  isInline: true
});
...  
//Add additional code
```

### ServerResponse.writeFile(options)

- **Method Description**: Method used to write line information to the response.
- **Returns**: Void
- **Supported Script Types**: Server-side scripts
- **Governance**: None
- **Module**: N/http Module
N/http Module

SuiteScript 2.0 API Reference

Parent Object
http.ServerResponse

Sibling Object Members
ServerResponse Object Members

Since
2015.2

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>options.output</td>
<td>string</td>
<td>required</td>
<td>The output string being written.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>(param name)</td>
<td>The value input for options.output is not a string.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
serverResponse.writeLine({
  output: 'this is a sample string'
});
...
//Add additional code
```

ServerResponse.writePage(options)

Method Description
Method used to generate a page.

Returns
Void

Supported Script Types
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/http Module

Parent Object
http.ServerResponse

Sibling Object Members
ServerResponse Object Members

Since
2015.2
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>options.pageObject</td>
<td>serverWidget.Assistant</td>
<td>required</td>
<td>A standalone page object in the form of an assistant, form or list.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument:</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>(param name)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/http Module Script Sample](#).

```javascript
//Add additional code
...
var myPageObj = serverWidget.createList({
  title: 'Simple List'
});

ServerResponse.writePage({
  pageObject: myPageObj
});
...
//Add additional code
```

ServerResponse.headers

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The server response headers.</td>
<td>Object</td>
</tr>
<tr>
<td></td>
<td>This property is read-only.</td>
<td></td>
</tr>
</tbody>
</table>

Supported Script Types

- Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

Module

- N/http Module

Parent Object

- [http.ServerResponse](#)
## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/http Module Script Sample](#).

```javascript
//Add additional code
...
if serverResponse.headers.Content-Type === 'text/plain'
return true

log.debug({
    title: 'Server Response Headers',
    details: serverResponse.headers
});
...
//Add additional code
```

### http.get(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send an HTTP GET request and return the response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>http.ClientResponse</td>
</tr>
</tbody>
</table>

**Supported Script Types**

- Server-side scripts
  
  For more information, see the help topic [SuiteScript 2.0 Script Types](#).

<table>
<thead>
<tr>
<th>Governance</th>
<th>10 units</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/http Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTP URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTP headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: {param name}</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/http Module Script Sample](#).

```javascript
//Add additional code
...
var response = http.get(
    {url: 'http://www.google.com'}
);
...
//Add additional code
```

http.get.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send an HTTP GET request asynchronously and return the response</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>For information about the parameters and errors thrown for this method, see <a href="#">http.get(options)</a>. For additional information on promises, see <a href="#">Promise Object</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td><a href="#">http.get(options)</a></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client-side scripts</td>
</tr>
<tr>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>10 units</td>
</tr>
<tr>
<td>Module</td>
<td><a href="#">N/http Module</a></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```javascript
//Add additional code
...
http.get.promise(
    {url: 'http://www.google.com'}
)
.then(function(response){
```
http.delete(options)

Method Description
Method used to send an HTTP DELETE request and return the response.

| Important: | If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs. |

| Note: | This method does not include an options.body parameter. Postdata is not required when the HTTP method is a DELETE request. |

Returns
http.ClientResponse

Supported Script Types
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 units

Module
N/http Module

Since
2015.2

Parameters

| Note: | The options parameter is a JavaScript object. |

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTP URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTP headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>
http.delete.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send an HTTP DELETE request asynchronously and return the response.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>For information about the parameters and errors thrown for this method, see http.delete(options). For additional information on promises, see Promise Object.</td>
</tr>
<tr>
<td>Returns</td>
<td>A http.ClientResponse object</td>
</tr>
<tr>
<td>Synchronous Version</td>
<td>http.delete(options)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
<tr>
<td>Governance</td>
<td>10 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/http Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
http.delete.promise({
  url: 'http://www.mytestwebsite.com'
});
...
//Add additional code
```

```javascript
//Add additional code
...
http.delete.promise({
  url: 'http://www.mytestwebsite.com'
}).then(function(response) {
  log.debug(
    {
      title: 'Response',
      details: response
    });
});
...catch(function onRejected(reason) {
```
```javascript
log.debug({
    title: 'Invalid Request: ',
    details: reason
});
```

## http.request(options)

Method used to send an HTTP request and return the response.

### Important:
If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.

### Returns
A `http.ClientResponse` object. For a complete list of this object's properties, see [ClientResponse Object Members](#).

### Supported Script Types
Server-side scripts
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

### Governance
10 units

### Module
N/http Module

### Since
2015.2

## Parameters

**Note:** The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.method</td>
<td><code>enum</code></td>
<td>required</td>
<td>The HTTP request method. Set using the <code>http.Method</code> enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.url</td>
<td><code>string</code></td>
<td>required</td>
<td>The HTTP URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.body</td>
<td><code>string</code></td>
<td>optional</td>
<td>The POST data if the method is <code>POST</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>or <code>object</code></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>options.headers</td>
<td><code>object</code></td>
<td>optional</td>
<td>An object containing request headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
var response = http.request({
  method: http.Method.GET,
  url: 'http://www.google.com'
});
...
//Add additional code
```

### Syntax

#### http.request.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send an HTTP request asynchronously and return the response.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>For information about the parameters and errors thrown for this method, see <code>http.request(options)</code>. For additional information on promises, see Promise Object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>A <code>http.ClientResponse</code> object</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Synchronous Version</th>
<th><code>http.request(options)</code></th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>All client-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Governance</td>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/http Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
http.request.promise({
  method: http.Method.GET,
  url: 'http://www.google.com'
}).then(function(response){
  log.debug({
    title: 'Response',
    details: response
  });
}).catch(function onRejected(reason) {
```

---

SuiteScript 2.0 API Reference
http.post(options)

Method Description
Method used to send an HTTP POST request and return the response.

Important: If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.

Returns
A `http.ClientResponse` object

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 units

Module
N/http Module

Since
2015.2

Parameters

Note: The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTP URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.body</td>
<td>string</td>
<td>object</td>
<td>required</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTP headers</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
```

//Add additional code
http.post.promise(options)

**Method Description**

Method used to send an HTTP POST request asynchronously and return the response.

**Note:** For information about the parameters and errors thrown for this method, see `http.post(options)`. For additional information on promises, see `Promise Object`.

**Returns**

A `http.ClientResponse` object

**Synchronous Version**

`http.post(options)`

**Supported Script Types**

All client-side scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](https://oracle.com).

**Governance**

10 units

**Module**

N/http Module

**Since**

2015.2

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see `Promise Object`.

```javascript
//Add additional code
...
http.post.promise({
    url: 'http://www.google.com',
    body: myPostDataObj
})
.then(function(response){
    log.debug({
        title: 'Response',
        details: response
    });
})
.catch(function onRejected(reason) {
    log.debug({
        title: 'Invalid Request: ',
        details: reason
    });
});
...```
http.put(options)

Method Description
Method used to send an HTTP PUT request and return the response.

**Important:** If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.

Returns  http.ClientResponse object

Supported Script Types  Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance  10 units

Module  N/http Module

Since  2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTP URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.body</td>
<td>string</td>
<td>required</td>
<td>The PUT data.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTP headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
var response = http.put({
  url: 'http://www.google.com',
  body: myDataObj,
};
```
http.put.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send an HTTP PUT request asynchronously and return the response.</th>
</tr>
</thead>
</table>

**Note:** For information about the parameters and errors thrown for this method, see `http.put(options)`. For additional information on promises, see `Promise Object`.

**Returns**

- `http.ClientResponse object`

**Synchronous Version**

- `http.put(options)`

**Supported Script Types**

- All client-side scripts
  
  For more information, see the help topic `SuiteScript 2.0 Client Script Type`.

**Governance**

- 10 units

**Module**

- `N/http Module`

**Since**

- 2015.2

**Syntax**

```javascript
//Add additional code
...
http.put.promise({
  url: 'http://www.google.com',
  body: myDataObj,
  headers: headerObj
})
  .then(function(response){
    log.debug({
      title: 'Response',
      details: response
    });
  })
  .catch(function onRejected(reason) {
    log.debug({
      title: 'Invalid Request: ',
      details: reason
    });
  })
...```

//Add additional code
http.CacheDuration

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported cache durations. This enum is used to set the value of the ServerResponse.setCdnCacheable(options) property.</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/http Module

**Values**
- LONG
- MEDIUM
- SHORT
- UNIQUE

**Syntax**

```javascript
//Add additional code
...
ServerResponse.setCdnCacheable({
  type: http.CacheDuration.MAX
});
...
//Add additional code
```

http.Method

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported HTTP requests. This enum is used to set the value of http.request(options) and ServerRequest.method.</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**
Server-side scripts
**Values**

- **DELETE**
- **GET**
- **HEAD**
- **PUT**
- **POST**

**Syntax**

```javascript
//Add additional code
...
var response = http.request({
    method: http.Method.GET,
    url: 'http://www.google.com'
});
...
//Add additional code
```

**http.RedirectType**

**Enum Description**

Holds the string values for supported NetSuite resources that you can redirect to. This enum is used to set the value of the `type` argument for `ServerResponse.sendRedirect(options)`.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/http Module

---

For more information, see the help topic SuiteScript 2.0 Script Types.
N/http Module

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
myServerResponseObj.sendRedirect({
  type: http.RedirectType.RECORD,
  identifier: record.Type.SALES_ORDER,
  parameters: {entity: 6}
});
```

N/https Module

Load the https module when you need to manage content sent to a third party via HTTPS calls. This module encapsulates all the functionality of the N/http Module, but does not allow the HTTP protocol. You can make HTTPS calls from client and server-side scripts.

SecureString functionality is supported only in server-side scripts. You can also use this functionality to perform various string transformations using methods that hash, encode, or append another string.

You can use this module to encode binary content or access a handle to the value in a NetSuite credential field.

When the https module is used, SuiteScript also loads the N/crypto Module and N/encode Module.

Important: Use TLS 1.2 for https requests. For more information, see the help topics FAQ: Transport Layer Security (TLS) Deprecations, specifically SuiteScript and TLS.

Important: NetSuite supports the same list of trusted third-party certificate authorities (CAs) as Microsoft. For a list of these CAs, see [http://social.technet.microsoft.com/wiki/contents/articles/31634.microsoft-trusted-root-certificate-program-participants-v-2016-april.aspx](http://social.technet.microsoft.com/wiki/contents/articles/31634.microsoft-trusted-root-certificate-program-participants-v-2016-april.aspx)

- N/https Module Members
- SecureString Object Members
- ClientResponse Object Members
- ServerResponse Object Members
- ServerRequest Object Members
- N/https Module Script Sample

General HTTPS Header Blacklist

Be aware that certain headers cannot be set manually when using https module methods. If a script attempts to set values for any of the following headers, the values are discarded. These headers are described in the following table.

- Connection
- Content-Length
- Host
- JSESSIONID
- Transfer-Encoding
- Upgrade
- Via
Suitelet Response HTTPS Header Blacklist

In addition to the headers described in General HTTPS Header Blacklist, certain headers cannot be set manually when interacting with the `https.ServerResponse` objects sent by Suitelets. If a script attempts to set values for any of these headers, the system throws an SSS_INVALID_HEADER error. These headers are described in the following table.

<table>
<thead>
<tr>
<th>Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access-Control-Allow-Origin</td>
<td></td>
</tr>
<tr>
<td>Allow</td>
<td></td>
</tr>
<tr>
<td>Connection</td>
<td></td>
</tr>
<tr>
<td>Content-Length</td>
<td></td>
</tr>
<tr>
<td>Content-Location</td>
<td></td>
</tr>
<tr>
<td>Content-MD5</td>
<td></td>
</tr>
<tr>
<td>Content-Range</td>
<td></td>
</tr>
<tr>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>JSESSIONID</td>
<td></td>
</tr>
<tr>
<td>Location</td>
<td></td>
</tr>
<tr>
<td>Proxy-Authenticate</td>
<td></td>
</tr>
<tr>
<td>Retry-After</td>
<td></td>
</tr>
<tr>
<td>Server</td>
<td></td>
</tr>
<tr>
<td>Trailer</td>
<td></td>
</tr>
<tr>
<td>Via</td>
<td></td>
</tr>
<tr>
<td>Warning</td>
<td></td>
</tr>
<tr>
<td>WWW-Authenticate</td>
<td></td>
</tr>
</tbody>
</table>

N/https Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td><code>https.SecureString</code></td>
<td><code>Object</code></td>
<td><code>Server-side scripts</code></td>
<td>Encapsulates data that may be sent to a third-party via an HTTPS call.</td>
</tr>
<tr>
<td></td>
<td><code>https.ClientResponse</code></td>
<td><code>read-only Object</code></td>
<td><code>Server-side scripts</code></td>
<td>Encapsulates the response to an HTTPS client request.</td>
</tr>
<tr>
<td></td>
<td><code>https.ServerRequest</code></td>
<td><code>read-only Object</code></td>
<td><code>Server-side scripts</code></td>
<td>Encapsulates the HTTPS request information sent to an HTTPS server. For example, a request received by a Suitelet or RESTlet.</td>
</tr>
<tr>
<td></td>
<td><code>https.ServerResponse</code></td>
<td><code>Object</code></td>
<td><code>Server-side scripts</code></td>
<td>Encapsulates the response from an HTTPS server to an HTTPS request. For example, a response from a Suitelet or RESTlet.</td>
</tr>
<tr>
<td>Method</td>
<td><code>https.createSecureKey(options)</code></td>
<td><code>Server-side scripts</code></td>
<td><code>Server-side scripts</code></td>
<td>Creates a key for the contents of a credential field.</td>
</tr>
<tr>
<td></td>
<td><code>https.createSecureKey.promise(options)</code></td>
<td><code>Server-side scripts</code></td>
<td><code>Server-side scripts</code></td>
<td>Creates a key asynchronously for the contents of a credential field.</td>
</tr>
<tr>
<td></td>
<td><code>https.delete(options)</code></td>
<td><code>Server-side scripts</code></td>
<td><code>Server-side scripts</code></td>
<td>Sends an HTTPS DELETE request and returns the response.</td>
</tr>
<tr>
<td></td>
<td><code>https.delete.promise(options)</code></td>
<td><code>Server-side scripts</code></td>
<td><code>Server-side scripts</code></td>
<td>Sends an HTTPS DELETE request asynchronously and returns the response.</td>
</tr>
<tr>
<td></td>
<td><code>https.get(options)</code></td>
<td><code>Server-side scripts</code></td>
<td><code>Server-side scripts</code></td>
<td>Sends an HTTPS GET request and returns the response.</td>
</tr>
<tr>
<td></td>
<td><code>https.get.promise(options)</code></td>
<td><code>Server-side scripts</code></td>
<td><code>Server-side scripts</code></td>
<td>Sends an HTTPS GET request asynchronously and returns the response.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>https.post(options)</td>
<td>https.â##ClientResponse</td>
<td>Server-side scripts</td>
<td>Sends an HTTPS POST request and returns the response.</td>
<td></td>
</tr>
<tr>
<td>https.â##post.â##promise(options)</td>
<td>https.â##ClientResponse</td>
<td>Client scripts</td>
<td>Sends an HTTPS POST request asynchronously and returns the response.</td>
<td></td>
</tr>
<tr>
<td>https.put(options)</td>
<td>https.â##ClientResponse</td>
<td>Server-side scripts</td>
<td>Sends an HTTPS PUT request and returns the response.</td>
<td></td>
</tr>
<tr>
<td>https.â##put.â##promise(options)</td>
<td>https.â##ClientResponse</td>
<td>Client scripts</td>
<td>Sends an HTTPS PUT asynchronously request and returns the response.</td>
<td></td>
</tr>
<tr>
<td>https.request(options)</td>
<td>https.â##ClientResponse</td>
<td>Server-side scripts</td>
<td>Sends an HTTPS request and returns the response. If a request fails, an error.SuiteScriptError is thrown.</td>
<td></td>
</tr>
<tr>
<td>https.â##request.â##promise(options)</td>
<td>https.â##ClientResponse</td>
<td>Client scripts</td>
<td>Sends an HTTPS request asynchronously and returns the response. If a request fails, a Promise.reject is thrown with a parameter Error.</td>
<td></td>
</tr>
</tbody>
</table>

**Enum**

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>https.CacheDuration</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported cache durations. This enum is used to set the value of the ServerResponse.â##setCdnCacheable(options) â##property.</td>
</tr>
<tr>
<td>https.Encoding</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported encoding types. This enum is used to set the value of parameters in SecureString.â##appendString(options), SecureString.â##convertEncoding(options), https.â##createSecureString(options).</td>
</tr>
<tr>
<td>https.HashAlg</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported hashing algorithms. This enum is used to set the value of parameters in SecureString.â##hash(options) and SecureString.â##hmac(options).</td>
</tr>
<tr>
<td>https.Method</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported HTTP requests. This enum is used to set the value of parameters in https.request(options) and to set the value of ServerRequest.method.</td>
</tr>
<tr>
<td>https.RedirectType</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported NetSuite resources to which you can redirect. This enum is used to set the value of parameters in ServerResponse.â##sendRedirect(options).</td>
</tr>
</tbody>
</table>

### SecureString Object Members

The following members are called on https.SecureString.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>

ClientResponse Object Members

The following members are called on http.ClientResponse.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>clientResponse.body</td>
<td>read-only string</td>
<td>Server-side scripts</td>
<td>The response body.</td>
</tr>
<tr>
<td></td>
<td>clientResponse.code</td>
<td>read-only number</td>
<td>Server-side scripts</td>
<td>The response code.</td>
</tr>
<tr>
<td></td>
<td>clientResponse.headers</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>The response body.</td>
</tr>
</tbody>
</table>

ServerRequest Object Members

The following members are called on the http.ServerRequest.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>serverRequest.#getLineCount(options)</td>
<td>number</td>
<td>Server-side scripts</td>
<td>Returns the number of lines in a sublist.</td>
</tr>
<tr>
<td></td>
<td>serverRequest.#getSublistValue(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns the value of a sublist line item.</td>
</tr>
<tr>
<td>Property</td>
<td>serverRequest.body</td>
<td>read-only string</td>
<td>Server-side scripts</td>
<td>The server request body</td>
</tr>
<tr>
<td></td>
<td>serverRequest.files</td>
<td>read-only string</td>
<td>Server-side scripts</td>
<td>The server request files.</td>
</tr>
<tr>
<td></td>
<td>serverRequest.headers</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>The server request headers.</td>
</tr>
<tr>
<td></td>
<td>serverRequest.method</td>
<td>https.Method enum</td>
<td>Server-side scripts</td>
<td>The HTTPS method for the server request.</td>
</tr>
<tr>
<td></td>
<td>serverRequest.#parameters</td>
<td>read-only Object</td>
<td>Server-side scripts</td>
<td>The server request parameters.</td>
</tr>
<tr>
<td></td>
<td>serverRequest.url</td>
<td>read-only string</td>
<td>Server-side scripts</td>
<td>The server request URL.</td>
</tr>
</tbody>
</table>

ServerResponse Object Members

The following members are called on the http.ServerResponse.
### N/https Module

#### Script Sample

The following example uses a GUID to generate a secure token and a secret key. Note this example is meant to show how to use the APIs but will not actually work in the debugger because the GUID does not exist in your account. Please try the Suitelet example for a more complete usage. To run, this sample in the debugger, you must replace the GUID with one specific to your account.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/runtime', 'N/https', 'N/crypto'],
  function(http, https, crypto) {
    function createSecureString() {
      var passwordGuid = '{284CFB2D225B1D76FB94D150207E49DF}';
      var secureToken = https.createSecureString({
        input: passwordGuid
      });
      var secretKey = https.createSecretKey({
        input: passwordGuid
      });
    }
  });
```

---

**Note:** This sample script uses a `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.
The following example is a Suitelet sample that shows creating a form field that generates a GUID.

For more information about credential fields, see `Form.addCredentialField(options)`.

**Note:** The default maximum length for a credential field is 32 characters. If needed, use the `Field.maxLength` property to change this value.

The values for `restrictToDomains`, `restrictToScriptIds`, and `baseUrl` in this sample are placeholders. You must replace them with valid values from your NetSuite account.

This sample uses the `define` function. The NetSuite Debugger cannot step through a `define` function. If you need to step through your code in the NetSuite Debugger, you must use a `require` function.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define(['N/ui/serverWidget', 'N/https', 'N/url'],
function(ui, https, url) {
    function onRequest(option) {
        if (option.request.method === 'GET') {
            var form = ui.createForm({
                title: 'Password Form'
            });
            var credField = form.addCredentialField({
                id: 'password',
                label: 'Password',
                restrictToDomains: ['system.netsuite.com'],
                restrictToCurrentUser: false,
                restrictToScriptIds: 'customscript_my_script'
            });
            credField.maxLength = 64;
            form.addSubmitButton();
            option.response.writePage({
                pageObject: form
            });
        }
        else {
            // Request to an existing suitelet with credentials
            var passwordGuid = option.request.parameters.password;
            //Replace SCRIPTID and DEPLOYMENTID with the internal ID of the suitelet script and deployment in your account
            var baseUrl = url.resolveScript({scriptID: SCRIPTID, deploymentID: DEPLOYMENTID, returnExternalURL: true});
            var authUrl = baseUrl + '?pwd=' + passwordGuid + ');
            var secureStringUrl = https.createSecureString({
                input: authUrl
            });
        }
    }
});
```
var secureStringPWD = https.createSecureString({
    input: '{' + passwordGuid + '}'
});

var headers = {
    'pwd': secureStringPWD
};

var response = https.get({
    credentials: [passwordGuid],
    url: secureStringUrl,
    headers: headers
});

}

return {
    onRequest: onRequest
};

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

https.SecureString

Object Description
Encapsulates a request string, such as a fragment of sensitive data that is going to be sent to a third party.

This object is needed when you create a securestring, put your data in it, and encode it a particular way.

For a complete list of this object’s methods, see SecureString Object Members.

Supported Script Types
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/https Module

Since
2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

//Add additional code
...
function createSecureString() {
    var passwordGuid = '{284CFB2D225B1D76FB94D159207E490F}';
    var secureToken = https.createSecureString({
        input: passwordGuid
    });
    ...
    //Add additional code
SecureString.appendSecureString(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to append a passed in https.SecureString to another https.SecureString.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>https.SecureString</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server-side scripts  
                      | For more information, see the help topic SuiteScript 2.0 Script Types.          |
| Governance         | None                                                                                |
| Module              | N/https Module                                                                     |
| Since               | 2015.2                                                                             |

Parameters

- **Note: The options parameter is a JavaScript object.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>

Syntax

- **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
string1.appendSecureString({
  secureString: secureString2
});
...
//Add additional code
```

SecureString appendString(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to append a passed string to an https.SecureString.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>https.SecureString</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server-side scripts  
                      | For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None                                                              |
| Module              | N/https Module                                                     |
| Since               | 2015.2                                                            |
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.input</td>
<td>string</td>
<td>required</td>
<td>The string to append.</td>
</tr>
<tr>
<td>options.inputEncoding</td>
<td>https.Encoding</td>
<td>required</td>
<td>The encoding of the string that is being appended.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
string1.appendString(
  input: '48656c6c6f20776f726c640d0a',
  encoding: https.Encoding.HEX);
...
//Add additional code
```

SecureString.convertEncoding(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Changes the encoding of a https.SecureString</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>https.SecureString</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.toEncoding</td>
<td>https.Encoding</td>
<td>required</td>
<td>The encoding to apply to the returned string.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code ...
https.convertEncoding({
  toEncoding: https.Encoding.HEX
});
```
SecureString.hash(options)

Method Description  Hashes an https.SecureString object
Returns  https.SecureString
Supported Script Types  Server-side scripts
Governance  None
Module  N/https Module
Since  2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
secureString = secureString.hash({
    algorithm: crypto.HashAlg.SHA256
});
...
//Add additional code
```

SecureString.hmac(options)

Method Description  Produces the securestring as an hmac.
Returns  https.SecureString
Supported Script Types  Server-side scripts
Governance  None
Module  N/https Module
Since  2015.2
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.key</td>
<td>crypto.SecretKey</td>
<td>required</td>
<td>A key returned from https.createSecureKey(options).</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
secureToken = secureToken.hmac(
  algorithm: crypto.HashAlg.SHA256,
  key: secretKey
);
...;
//Add additional code
```

### https.createSecureKey(options)

**Method Description**

Creates and returns a crypto.SecretKey object. This method can take a GUID. Use Form.addCredentialField(options) to generate a value.

You can put the key in your secure string. SuiteScript decrypts the value (key) and sends it to the server.

**Returns**

crypto.SecretKey

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/https Module

**Since**

2015.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.encoding</td>
<td>https.Encoding enum</td>
<td>optional</td>
<td>Specifies the encoding for the SecureKey.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### 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>options.guid</td>
<td>string</td>
<td>required</td>
<td>A GUID used to generate a secret key.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The GUID can resolve to either data or metadata.</td>
<td></td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
var secretKey = https.createSecretKey({
  encoding: https.Encoding.HEX,
  guid: '284CFB2D225B1D76FB94D150207E49DF'
});
...
//Add additional code
```

### https.createSecureKey.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates and returns a crypto.SecretKey object asynchronously.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>For information about the parameters and errors thrown for this method, see https.createSecureKey(options). For additional information on promises, see Promise Object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>A crypto.SecretKey object</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Synchronous Version</strong></td>
<td>https.createSecureKey(options)</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All client-side scripts</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/https Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
var secretKey = https.createSecureKey.promise({
  encoding: https.Encoding.HEX,
  guid: '284CFB2D225B1D76FB94D150207E49DF'
});
...
//Add additional code
```
**https.createSecureString(options)**

**Method Description**
Creates and returns an `https.SecureString`.

**Returns**
`https.SecureString`

**Supported Script Types**
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/https Module

**Since**
2015.2

**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>options.input</td>
<td>string</td>
<td>required</td>
<td>The string to convert to a securestring.</td>
<td>Release 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Version 2</td>
</tr>
<tr>
<td>options.##inputEncoding</td>
<td><code>https.Encoding</code> enum</td>
<td>optional</td>
<td>Identifies the encoding that the input string uses.</td>
<td>Release 15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Version 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default value is <code>UTF_8</code>.</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
var secureToken = https.createSecureString({
   input: passwordGuid
});
...
//Add additional code
```

**https.createSecureString.promise(options)**

**Method Description**
Creates and returns an `https.SecureString` asynchronously.
**Note:** For information about the parameters and errors thrown for this method, see `https.createSecureString(options)`. For additional information on promises, see `Promise Object`.

<table>
<thead>
<tr>
<th>Returns</th>
<th>SecureTokenResolver</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td><code>https.createSecureString(options)</code></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
var secureToken = https.createSecureString.promise({
  input: passwordGuid
});
...
// Add additional code
```

**https.ClientResponse**

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the response to an HTTPS client request.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This object is read-only.</td>
</tr>
<tr>
<td></td>
<td>For a complete list of this object's properties, see <code>ClientResponse Object Members</code>.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
var clientResponse = https.get({
```
url: 'https://www.testwebsite.com'  
});  
...  
//Add additional code

### ClientResponse.body

| Property Description | The client response body.  
This property is read-only. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |

**Module**  
N/https Module  

**Since**  
2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

### Syntax

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...  
var response = https.get({
  url: 'https://www.testwebsite.com'
});  
log.debug({
  title: 'Client Response Body',
  details: https.response.body
});  
...  
//Add additional code
```

### ClientResponse.code

| Property Description | The client response code.  
This property is read-only. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |
Module
N/https Module

Since
2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
...
var response = https.get({
  url: 'https://www.testwebsite.com'
});
log.debug({
  title: 'Client Response Code',
  details: https.response.code
});
...
//Add additional code
```

ClientResponse.headers

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The response header or headers.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is read-only.</td>
</tr>
</tbody>
</table>

Type
object

Supported Script Types
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/https Module

Since
2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/http Module Script Sample.

```javascript
//Add additional code
```
```javascript
var response = https.get({
    url: 'https://www.testwebsite.com'
});
log.debug({
    title: 'Client Response Header',
    details: https.response.headers
});
...  
//Add additional code
```

### https.ServerRequest

**Object Description**
Encapsulates the incoming https request information for an HTTPS server.  
This object is read-only.  
For a complete list of this object's methods and properties, see [ServerRequest Object Members](#).

**Supported Script Types**
Server-side scripts  
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/https Module

**Since**
2015.2

### Syntax

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [https.ServerRequest](#).

```javascript
//Add additional code
...  
serverRequest.getLineCount({
    group: 'sublistId'
});
...  
//Add additional code
```

### ServerRequest.getLineCount(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to return the number of lines in a sublist.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td>The number of lines in a sublist as a number.</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/https Module</td>
</tr>
</tbody>
</table>
Since 2015.2

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>options.group</td>
<td>string</td>
<td>required</td>
<td>The sublist internal ID.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverRequest.getLineCount({
  group: 'sublistId'
});
...
//Add additional code
```

ServerRequest.getSublistValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to return the value of a sublist line item.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>The value of the sublist line item as a string.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

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>options.group</td>
<td>string</td>
<td>required</td>
<td>The sublist internal ID.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.name</td>
<td>string</td>
<td>required</td>
<td>The name of the field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>string</td>
<td>required</td>
<td>The sublist line number.</td>
<td>2015.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>---------------------</td>
<td>-------------</td>
<td>-------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> Sublist index starts at 0.</td>
<td></td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverRequest.getSublistValue({
    group: 'item',
    name: 'amount',
    line: '2'
});
...
//Add additional code
```

**ServerRequest.body**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The server request body.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is read-only.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
</table>

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/https Module

**Since**

2015.2

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
```
ServerRequest.files

**Property Description**
The server request files.
This property is read-only.

**Type**
Object

**Supported Script Types**
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/https Module

**Since**
2015.2

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
log.debug({
    title: 'Server Request Body',
    details: https.request.body
});
...
//Add additional code

var file = request.files['file_id'];
```

ServerRequest.headers

**Property Description**
This object represents a series of key/value pairs. Each pair represents a server request header name and its value.

Typically, this object encapsulates two iterations of each header name: one in lower case and another in title case. This behavior is designed so that you can use either lower case or title.
case when you reference a header. However, the existence of title-case iterations of header names is not guaranteed. For best results, refer to header names using all lower-case letters (and hyphens, when applicable).

This property is read-only.

**Important:** The server request headers and their values are subject to change. If you use these headers in your scripts, you are responsible for testing them to make sure that they contain the information you need. For example, when making an HTTP call to a Suitelet, some headers might be filtered out. Filtering can occur if the headers affect how NetSuite processes the request internally. These filtered headers are not available to the Suitelet, so you should test to see whether a header was filtered out. If so, use a different header instead.

**Type**

Object

**Supported Script Types**

Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/https Module

**Since**

2015.2

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/https Module Script Sample](#).

```javascript
//Add additional code
...
log.debug({
  title: 'Server Request Headers',
  details: https.request.headers
});
...
//Add additional code
```

**ServerRequest.method**

**Property Description**

The server request https method.

This property is read-only.

**Type**

enum

**Supported Script Types**

Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/https Module
### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see Parameters.

```javascript
//Add additional code
...
log.debug({
  title: 'Server Request Method',
  details: https.request.method
});
...
//Add additional code
```

### ServerRequest.parameters

**Property Description**

The server request parameters. This property is read-only.

**Type**

Object

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/https Module

**Since**

2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td></td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
log.debug({
  title: 'Server Request Parameters',
  details: https.request.method
});
```
ServerRequest.url

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The server request URL. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
log.debug(
    {
        title: 'Server Request URL',
        details: https.request.url
    });
...//Add additional code
```

https.ServerResponse

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the response to an incoming http request from an HTTP server. For example, a response from a Suitelet or RESTlet.</th>
</tr>
</thead>
<tbody>
<tr>
<td>For a complete list of this object's methods and properties, see ServerResponse Object Members.</td>
<td></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
</tbody>
</table>
Since 2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
//Add additional code
```

Method Description

Method used to add a header to the response.

If the same header has already been set, this method adds another line for that header. For example:

```
Vary: 'Accept-Language'
Vary: 'Accept-Encoding'
```

Returns

Void

Supported Script Types

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/https Module

Since 2015.2

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>options.name</td>
<td>string</td>
<td>required</td>
<td>The name of the header.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>required</td>
<td>The value used to set the header.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>SSS_INVALID_HEADER</td>
<td>One or more headers are not valid.</td>
<td>The header name or value is invalid.</td>
</tr>
</tbody>
</table>
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverResponse.addHeader({
    name: 'Accept-Language',
    value: 'en-us',
});
...
//Add additional code
```

**ServerResponse.getHeader**(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to return the value or values of a response header. If multiple values are assigned to the header name, the values are returned as an Array.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**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>options.name</td>
<td>string</td>
<td>required</td>
<td>The name of the header.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverResponse.getHeader({
    name: 'Accept-Language'
});
```
ServerResponse.sendRedirect(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to create a redirect URL that resolves to a NetSuite resource. For example, you could use this method to redirect to a new sales order page for a particular entity.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

**Important:** All parameters must be prefixed with custparam.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The type of resource to which the script redirects. Use the https.RedirectType enum to set a value for this parameter.</td>
</tr>
<tr>
<td>options.identifier</td>
<td>number</td>
<td>required</td>
<td>The primary ID for this resource. The value you use varies depending on the value of options.type, as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- MEDIA_ITEM — Use the internal ID of a file stored in the NetSuite File Cabinet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- RECORD — Use the record.Type enum to identify the appropriate record type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- RESTLET — Use the script ID from the script record of the appropriate RESTlet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- SUITELET — Use the script ID from the script record of the appropriate Suitelet.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- TASK_LINK — Use the appropriate Task ID. Supported IDs are listed in Task IDs.</td>
</tr>
<tr>
<td>options.id</td>
<td>string</td>
<td>optional</td>
<td>The secondary ID for this resource. If the options.type parameter is set to SUITELET or RESTLET, use the deployment ID. If the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>options.type parameter is set to RECORD, you can use the internal ID of a specific record instance.</td>
</tr>
</tbody>
</table>
### N/https Module

#### SuiteScript 2.0 API Reference

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
</table>
| options.editMode| boolean| optional            | Applicable when redirecting to a record. Use the following values:  
- **true** — returns the record in edit mode.  
- **false** — returns the record in view mode.  
The default value is false. | 2015.2 |
| options.parameters| object  | optional            | Additional URL parameters as key-value pairs.                                                                                                                                                | 2015.2 |

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is missing. Note that this error is thrown if the script includes a typo in certain enums. For example, you see this error if you use <code>https.RedirectType.TASKLINK</code> instead of <code>https.RedirectType.TASK_LINK</code> in the options.type field.</td>
</tr>
<tr>
<td>SSS_INVALID_URL_CATEGORY</td>
<td>The options.type: (type) is not valid. Please use the RedirectType enum for supported types.</td>
<td>The script uses an unrecognized string value for the options.type parameter. To avoid this error, use the <code>https.RedirectType</code> enum.</td>
</tr>
<tr>
<td>INVALID_TASK_ID</td>
<td>The task ID: (id) is not valid. Please refer to the documentation for a list of supported task IDs.</td>
<td>The options.type parameter is set to TASK_LINK, and the script uses an invalid task ID for options.identifier. For a list of valid IDs, see the help topic Task IDs.</td>
</tr>
<tr>
<td>INVALID_RCRD_TYPE</td>
<td>The record type (type) is invalid.</td>
<td>The options.type parameter is set to RECORD, and the script uses an unrecognizable string value for options.identifier. To avoid this error, use the <code>record.Type</code> enum to identify the appropriate record type.</td>
</tr>
<tr>
<td>INVALID_ID</td>
<td>You have provided an invalid script id or internal id: (id)</td>
<td>The options.type parameter is set to RESTLET or SUITELET, and the script uses an invalid ID for options.identifier or options.id.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
myServerResponseObj.sendRedirect({
  type: https.RedirectType.RECORD,
  identifier: record.Type.SALES_ORDER,
  parameters: {entity: 8}
});
...
//Add additional code
```
ServerResponse.setHeader(options)

**Method Description**  
Method used to set the value of a response header.

**Returns**  
Void

**Supported Script Types**  
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**  
None

**Module**  
N/https Module

**Since**  
2015.2

### 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>options.name</td>
<td>string</td>
<td>required</td>
<td>The name of the header.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>required</td>
<td>The value used to set the header.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>SSS_INVALID_HEADER</td>
<td>One or more headers are not valid.</td>
<td>The header name or value is invalid.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
serverResponse.setHeader({
    name: ‘Accept-Language’,
    value: ‘en-us’,
});
...
// Add additional code
```

ServerResponse.renderPdf(options)

**Method Description**  
Method used to generates and renders a PDF directly to the response.

**Returns**  
Void

**Supported Script Types**  
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.
N/https Module

Governance

10 units

Module

N/https Module

Since

2015.2

Parameters
Parameter

Type

Required / Optional

Description

Since

options.xmlString

string

required

Content of the pdf.

2015.2

Errors
Error Code

Message

Thrown If

SSS_MISSING_REQD_ARGUMENT

Missing a required argument:
{param name}

A required parameter is not passed.

Syntax
Important: The following code snippet shows the syntax for this member. It is not a

functional example. For a complete script example, see N/https Module Script Sample.
//Add additional code
...
serverResponse.renderPDF({
xmlString:'<?xml version="1.0"?>\n<!DOCTYPE pdf PUBLIC "-//big.faceless.org//report" "report-1.1.dtd">\n<pdf>\n<body
font-size="18">\nHello World!\n</body>\n</pdf>'
});
...
//Add additional code

ServerResponse.setCdnCacheable(options)
Method Description

Method used to set CDN caching for a period of time.

Returns

Void

Supported Script Types

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/https Module

Since

2015.2

Parameters
Parameter

Type

Required / Optional

Description

Since

options.type

string

required

The value of the caching duration. Set using
the https.CacheDuration.

2015.2

SuiteScript 2.0 API Reference

333


Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: {param name}</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverResponse.setCdnCacheable({
  type: https.CacheDuration.MAX
});
...
//Add additional code
```

ServerResponse.write(options)

Method Description

Method used to write information to the response.

Note: This method accepts only strings. To pass in a file, you can use ServerResponse.writeFile(options).

Returns

Void

Supported Script Types

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/https Module

Since

2015.2

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>options.output</td>
<td>string</td>
<td>required</td>
<td>The output string being written.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: {param name}</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>
ServerResponse.writeFile(options)

Method Description: Method used to write a file to the response.

Returns: Void

Supported Script Types: Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance: None

Module: N/https Module

Since: 2015.2

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>options.file</td>
<td>file.File</td>
<td>required</td>
<td>A file.File Object that encapsulates the file to be written.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.isInline</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>Determines whether the field is inline. If true, the file is inline.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>(param name)</td>
<td>The value input for options.file is not a file.File Object.</td>
</tr>
</tbody>
</table>
Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverResponse.writeFile({
    file: myFileObj,
    isInline: true
});
...
//Add additional code
```

ServerResponse.writeLine(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to write line information to the response.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

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>options.output</td>
<td>string</td>
<td>required</td>
<td>The output string being written.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>(param name)</td>
<td>The value input for options.output is not a string.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
```
ServerResponse.writePage(options)

Method Description  Method used to generate a page.
Returns  Void
Supported Script Types  Server-side scripts
Governance  None
Module  N/https Module
Since  2015.2

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>options.pageObject</td>
<td>serverWidget.Assistant</td>
<td>required</td>
<td>A standalone page object in the form of an assistant, form or list.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
serverResponse.writeLine({
    output: 'this is a sample string'
});
...
//Add additional code

var myPageObj = serverWidget.createList({
    title: 'Simple List'
});

ServerResponse.writePage({
    pageObject: myPageObj
});
```
ServerResponse.headers

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The server response headers. This property is read-only.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Object</td>
</tr>
<tr>
<td>Notes</td>
<td>Note that if multiple values are assigned to one header name, the values are returned as an array. For example:</td>
</tr>
<tr>
<td></td>
<td><code>{Vary: ['Accept-Language', 'Accept-Encoding']}</code></td>
</tr>
</tbody>
</table>

Supported Script Types

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/https Module

Since

2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>You attempt to edit this property.</td>
</tr>
</tbody>
</table>

Syntax

```
//Add additional code
...
if serverResponse.headers.Content-Type === 'text/plain'
return true

log.debug({
  title: 'Server Response Headers',
  details: serverResponse.headers
});
...
//Add additional code
```

https.get(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to send an HTTPS GET request and return the response</th>
</tr>
</thead>
</table>
Returns

https.ClientResponse

Supported Script Types

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

10 units

Module

N/https Module

Since

2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTPS URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTPS headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
var response = https.get({
    url: 'https://www.testwebsite.com'
});
...
//Add additional code
```

https.get.promise(options)

Method Description

Method used to send an HTTPS GET request asynchronously and return the response

Note: For information about the parameters and errors thrown for this method, see https.get(options). For additional information on promises, see Promise Object.

Returns

A https.ClientResponse object.
**https.get(options)**

**Supported Script Types**  
All client-side scripts  
For more information, see the help topic [SuiteScript 2.0 Client Script Type](https://oracle netsuite.com).

**Governance**  
10 units

**Module**  
N/https Module

**Since**  
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](https://oracle netsuite.com).

```javascript
//Add additional code
...
https.get.promise({
    url: 'https://www.testwebsite.com'
})
.then(function(response){
    log.debug(
        {
            title: 'Response',
            details: response
        }
    )
})
.catch(function onRejected(reason) {
    log.debug(
        {
            title: 'Invalid Get Request: ',
            details: reason
        }
    )
})
...
//Add additional code
```

### https.delete(options)

**Method Description**  
Method used to send an HTTPS DELETE request and returns the response.

⚠️ **Important:** If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.

**Note:** This method does not include an `options.body` parameter. Postdata is not required when the HTTPS method is a DELETE request.

**Returns**  
https.ClientResponse

**Supported Script Types**  
Server-side scripts  
For more information, see the help topic [SuiteScript 2.0 Script Types](https://oracle netsuite.com).
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTPS URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTPS headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/https Module Script Sample](#).

```javascript
//Add additional code
...
var response = https.delete({
  url: 'http://www.mytestwebsite.com'
});
...
//Add additional code
```

`https.delete.promise(options)`

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method used to send an HTTP DELETE request asynchronously and return the response.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For information about the parameters and errors thrown for this method, see `https.delete(options)`. For additional information on promises, see [Promise Object](#).
**Supported Script Types**  
All client-side scripts  
For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**  
10 units

**Module**  
N/https Module

**Since**  
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
https.delete.promise({
  url: 'http://www.mytestwebsite.com'
})
  .then(function(response){
    log.debug({
      title: 'Response',
      details: response
    });
  })
  .catch(function onRejected(reason) {
    log.debug({
      title: 'Invalid Request: ',
      details: reason
    });
  })
...
//Add additional code
```

**https.request(options)**

<table>
<thead>
<tr>
<th><strong>Method Description</strong></th>
<th>Method used to send an HTTPS request and return the response.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Important:</strong></td>
<td>If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Returns</strong></th>
<th>An https.ClientResponse Object. For a complete list of this object’s properties, see ClientResponse Object Members.</th>
</tr>
</thead>
</table>
| **Supported Script Types** | Server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| **Governance** | 10 units |
| **Module** | N/https Module |
| **Since** | 2015.2 |
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.method</td>
<td>enum</td>
<td>required</td>
<td>The HTTPS request method. Set using the <code>https.Method</code> enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTPS URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.body</td>
<td>string</td>
<td>optional</td>
<td>The POST data if the method is POST.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If the method is <code>DELETE</code>, this body data is ignored.</td>
<td></td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>An object containing request headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
var response = https.request({
  method: https.Method.GET,
  url: 'https://www.testwebsite.com'
});
...
//Add additional code
```

`https.request.promise(options)`

Method Description

Method used to send an HTTP request asynchronously and return the response.

Note: For information about the parameters and errors thrown for this method, see `https.request(options)`. For additional information on promises, see `Promise Object`.

Returns

A `https.ClientResponse` object
### https.request(options)

**Synchronous Version**

https.request(options)

**Supported Script Types**

All client-side scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**

10 units

**Module**

N/https Module

**Since**

2015.2

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
https.request.promise({
  method: https.Method.GET,
  url: 'https://www.testwebsite.com'
})
.then(function(response){
  log.debug(
    {
      title: 'Response',
      details: response
    });

})
.catch(function onRejected(reason) {
  log.debug(
    {
      title: 'Invalid Request: ',
      details: reason
    });

})
...
//Add additional code
```

### https.post(options)

**Method Description**

Method used to send an HTTPS POST request and return the response.

⚠️ **Important:** If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.

**Returns**

An https.ClientResponse Object

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

10 units

**Module**

N/https Module
N/https Module

SuiteScript 2.0 API Reference

Since 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTPS URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.body</td>
<td>string</td>
<td>object</td>
<td>The POST data.</td>
<td></td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTPS headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: {param name}</td>
<td>A required parameter is not passed.</td>
</tr>
<tr>
<td>SSS_REQUEST_LOOP_DETECTED</td>
<td>script executes a recursive function that has exceeded the limit for the number of times a script can call itself using an HTTP request. Please examine the script for a potential infinite recursion problem.</td>
<td>A script is calling back into itself recursively via an http/https request.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
var response = https.post({
    url: 'https://www.testwebsite.com',
    body: myPostDataObj
});
...
//Add additional code
```

https.post.promise(options)

Method Description

Method used to send an HTTPS POST request asynchronously and return the response.

Note: For information about the parameters and errors thrown for this method, see https.post(options). For additional information on promises, see Promise Object.

Returns

A https.ClientResponse object

Synchronous Version

https.post(options)
Supported Script Types  All client-side scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance  10 units

Module  N/https Module

Since  2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
https.post.promise({
  url: 'https://www.testwebsite.com',
  body: myPostDataObj
})
  .then(function(response){
    log.debug({
      title: 'Response',
      details: response
    });
  })
  .catch(function onRejected(reason) {
    log.debug({
      title: 'Invalid Request: ',
      details: reason
    });
  })
...
//Add additional code
```

### https.put(options)

**Method Description**  Method used to send an HTTPS PUT request and return server response.

⚠️ **Important:** If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.

**Returns**  An `https.ClientResponse` Object

**Supported Script Types**  Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance  10 units

Module  N/https Module

Since  2015.2
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The HTTPS URL being requested</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.body</td>
<td>string</td>
<td>object</td>
<td>The PUT data.</td>
<td></td>
</tr>
<tr>
<td>options.headers</td>
<td>object</td>
<td>optional</td>
<td>The HTTPS headers.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Missing a required argument: (param name)</td>
<td>A required parameter is not passed.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
var response = https.put({
  url: 'https://www.testwebsite.com',
  body: myDataObj,
  headers: headerObj
});
...
//Add additional code
```

https.put.promise(options)

**Method Description**

Method used to send an HTTPS PUT request asynchronously and return the response.

**Note:** For information about the parameters and errors thrown for this method, see https.put(options). For additional information on promises, see Promise Object.

<table>
<thead>
<tr>
<th>Returns</th>
<th>https.ClientResponse object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td>https.put(options)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>10 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/https Module</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
// Add additional code
...
https.put.promise(
    url: 'https://www.testwebsite.com',
    body: myDataObj,
    headers: headerObj
)
  .then(function(response){
    log.debug({
      title: 'Response',
      details: response
    });
  })
  .catch(function onRejected(reason) {
    log.debug({
      title: 'Invalid Request: ',
      details: reason
    });
  })
...
// Add additional code
```

### https.CacheDuration

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported cache durations. This enum is used to set the value of the <code>ServerResponse.setCdnCacheable(options)</code> property.</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>LONG</td>
</tr>
<tr>
<td>MEDIUM</td>
</tr>
<tr>
<td>SHORT</td>
</tr>
<tr>
<td>UNIQUE</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/https Module Script Sample](#).

```javascript
// Add additional code
...
ServerResponse.setCdnCacheable({
  type: https.CacheDuration.MAX
});
...
// Add additional code
```

### https.Encoding

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported encoding values.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Module</strong></td>
<td>N/https Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>- UTF_8</td>
</tr>
<tr>
<td>- BASE_16</td>
</tr>
<tr>
<td>- BASE_32</td>
</tr>
<tr>
<td>- BASE_64</td>
</tr>
<tr>
<td>- BASE_64_URL_SAFE</td>
</tr>
<tr>
<td>- HEX</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/https Module Script Sample](#).

```javascript
// Add additional code
...
var mySecretKey = https.createSecretKey({
  encoding: https.Encoding.HEX,
  guid: '284CFB20225B1D74FB94D0150207E490F'
});
...
// Add additional code
```
https.HashAlg

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported hashing algorithms.</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/https Module

**Values**

- SHA1
- SHA256
- SHA512
- MD5

**Syntax**

```javascript
// Add additional code
...
var mySecureString = https.createServerString({
    input: 'ConvertMe'
});
var mySecureStringHash = mySecureString.hash({
    algorithm: https.HashAlg.SHA256
});
...
// Add additional code
```

https.Method

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported HTTPS requests. This enum is used to set the value of https.request(options) and ServerRequest.method.</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

### Module

**N/https Module**

### Values

- DELETE
- GET
- HEAD
- PUT
- POST

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/https Module Script Sample`.

```javascript
//Add additional code
...
var response = https.request({
    method: https.Method.GET,
    url: 'https://www.testwebsite.com'
});
...
//Add additional code
```

### https.RedirectType

**Enum Description**

Holds the string values for supported NetSuite resources to which you can redirect. This enum is used to set the value of the `type` argument for `ServerResponse.sendRedirect(options)`.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

**N/https Module**

### Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEDIA_ITEM</td>
<td>A file in the NetSuite File Cabinet</td>
</tr>
<tr>
<td>RECORD</td>
<td>A NetSuite record.</td>
</tr>
</tbody>
</table>
### N/https Module

**Value** | **Description**
--- | ---
RESTLET | A deployed RESTlet.
SUITELET | A deployed Suitelet.
TASK_LINK | A page in NetSuite, as defined by a valid Task ID.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/https Module Script Sample.

```javascript
//Add additional code
...
myServerResponseObj.sendRedirect({
  type: https.RedirectType.RECORD,
  identifier: record.Type.SALES_ORDER,
  parameters: {entity: 6}
});
//Add additional code
```

### N/https/clientCertificate Module

Load the clientCertificate module to send SSL requests with a digital certificate.

#### Note

Only POST is supported for NetSuite Version 19.1.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
</table>

### N/https/clientCertificate Module Script Sample

The following is an example of how to send a certificate to a Brazilian tax authority for authentication.

**Note:** This sample script uses a `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics [SuiteScript 2.0 Script Basics](https://developers.netsuite.com/) and [SuiteScript 2.0 Script Types](https://developers.netsuite.com/).

```javascript
/**
 * @NApiVersion 2.x
 */
require(["N/https/clientCertificate"],
    function (cert) {
```
var url = "https://nfe.fazenda.sp.gov.br/ws/cadconsultacadastro4.asmx";


var key = "custcertificate1";
var headers = {
    "Content-Type": "application/soap+xml"
};

var response = cert.post({
    url: url,
    certId: key,
    body: data,
    headers: headers
});

log.debug(response.body);

---

clientCertificate.post(options)

**Method Description**

Method used to send a SSL secured request to a remote service and return the response.

**Important:** If negotiating a connection to the destination server exceeds 5 seconds, a connection timeout occurs. If transferring a payload to the server exceeds 45 seconds, a request timeout occurs.

**Returns**

An https.ClientResponse Object

**Supported Script Types**

Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

10 units

**Module**

N/https/clientCertificate Module

**Since**

2019.1

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required/Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The URL address of the remote server.</td>
<td>2019.1</td>
</tr>
<tr>
<td>options.certId</td>
<td>string</td>
<td>required</td>
<td>The ID of the client certificate.</td>
<td>2019.1</td>
</tr>
<tr>
<td>options.body</td>
<td>string</td>
<td>required</td>
<td>The POST data to be sent to the remote server.</td>
<td>2019.1</td>
</tr>
</tbody>
</table>
N/log Module

Use the log module to access methods for logging script execution details.

The log methods can be accessed globally or by loading this module. Load the N/log module when you want to manually access its members, such as for testing purposes. For more information about global objects, see SuiteScript 2.0 Global Objects.

- N/log Module Members
- N/log Module Guidelines
- Using Log Levels
- Viewing Script Execution Logs
- log Module Script Sample
- Governance on Script Logging

N/log Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>log.audit(options)</td>
<td>Void</td>
<td>Client and server-side scripts</td>
<td>Logs an entry of type AUDIT to the Execution Log tab of the script deployment for the current script.</td>
</tr>
<tr>
<td></td>
<td>log.debug(options)</td>
<td>Void</td>
<td>Client and server-side scripts</td>
<td>Logs an entry of type DEBUG to the Execution Log tab of the script deployment for the current script.</td>
</tr>
<tr>
<td></td>
<td>log.â##emergencyâ##(options)</td>
<td>Void</td>
<td>Client and server-side scripts</td>
<td>Logs an entry of type EMERGENCY to the Execution Log tab of the script deployment for the current script.</td>
</tr>
<tr>
<td></td>
<td>log.error(options)</td>
<td>Void</td>
<td>Client and server-side scripts</td>
<td>Logs an entry of type ERROR to the Execution Log tab of the script deployment for the current script.</td>
</tr>
</tbody>
</table>

N/log Module Guidelines

- NetSuite governs the amount of logging that can be done in any specific 60 minute time period. A company is allowed to make up to 100,000 log object method calls across all of their scripts. Script owners are notified if NetSuite detects that one script is logging excessively and automatically adjusts the log level.
- NetSuite purges system errors older than 60 days and user-generated logs older than 30 days. Because log persistence is not guaranteed, NetSuite recommends using custom records if you want to store script execution logs for extended periods.
- The Execution Log tab also lists notes returned by NetSuite such as error messages. For more information, see N/error Module.
If you deploy a client script to a form using `Form.clientScriptFileId` or `Form.clientScriptModulePath`, using the N/log module adds the logs to the deployment of the parent script. The parent script can be either a beforeLoad user event script or a SuiteScript 2.0 Suitelet Script Type.

When an object (that is not a string) is passed to a log object method, NetSuite runs `JSON.stringify(obj)` on any values that are passed as the details parameter and equal a JavaScript object.

```javascript
... // log.debug(rec) //Shows the JSON representation of the current values in a record object
  var id = rec.save();
  ...
```

Using Log Levels

Use the log methods along with the Log Level field on the Script Deployment to determine whether to log an entry on the Execution Log subtab. If a log level is defined on a Script Deployment, then only log Object method calls with a log type equal to or greater than this log level will be logged. This is useful during the debugging of a script or for providing useful execution notes for auditing or tracking purposes.

Log levels and log Object methods act as a filter on the amount of information logged. The following log levels are supported:

<table>
<thead>
<tr>
<th>Log Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debug</td>
<td>Shows all Audit, Error, and Emergency information on the Execution Log tab. This type of logging is suitable only for testing scripts. To avoid excessive logging, the debug log level is not recommended for active scripts in production.</td>
</tr>
<tr>
<td>Audit</td>
<td>Shows a record of events that have occurred during the processing of the script (for example, “A request was made to an external site.”).</td>
</tr>
<tr>
<td>Error</td>
<td>Shows only unexpected script errors.</td>
</tr>
<tr>
<td>Emergency</td>
<td>Shows only the most critical errors in the script log.</td>
</tr>
</tbody>
</table>

Viewing Script Execution Logs

To view logs for a specific script, see the Execution Log subtab of a Script Deployment record. These logs are not guaranteed to persist for 30 days and may be purged to enhance performance if volume is high.

To view script execution log details for various scripts, go to Customization > Scripting > Script Execution Logs Customization > Scripting > Script Execution Logs. This list of script execution logs is an enhanced repository that stores all log details for 30 days.

On this page, you can perform the following tasks:

- Search for specific logs using filter options, such as log level, execution date range, and script name.

  **Note:** The log list shows 10,000 entries at a time for a given filtered criteria. Users can view other logs by using the Date and Script filter options.

- Download the list as a CSV file or an Excel spreadsheet.
- Print the list.

**log Module Script Sample**

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/log'],
function(myLog) {
    var myObject = {
        name: 'Jane',
        id: '123'
    }
    myLog.debug({
        title: 'hello!'
    });
    myLog.debug({
        title: 'hello!',
        details: 'world'
    });
    myLog.debug({
        title: 'myObj',
        details: myObject
    });
});
```

**log.audit(options)**

| Method Description | Logs an entry of type AUDIT to the Execution Log tab of the script deployment for the current script. This entry will not appear on the Execution Log tab if the Log Level field for the script deployment is set to ERROR or above. Use this method for scripts in production. |
| Supported Script Types | All script types For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance | Amount of logging in any 60 minute period is limited. See N/log Module Guidelines. |
| Module | N/log Module |
| Since | 2016.1 |
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>string</td>
<td>Optional</td>
<td>String to appear in the Title column on the Execution Log tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (&quot;&quot;), or omit it, the word “Untitled” appears for the log entry.</td>
<td>2016.1</td>
</tr>
<tr>
<td>details</td>
<td>any</td>
<td>Required</td>
<td>You can pass any value for this parameter. If the value is a JavaScript Object type, JSON.stringify(obj) is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

The following code snippet shows the syntax for this method.

```javascript
//Add additional code
...
var var1 = 'value';
log.audit({
    title: 'Audit Entry',
    details: 'Value of var1 is: ' + var1
});
...
//Add additional code
```

**log.debug**

**Method Description**

Logs an entry of type DEBUG to the Execution Log tab of the script deployment for the current script.

This entry does not appear on the Execution Log tab if the Log Level field for the script deployment is set to AUDIT or above.

Use this method for scripts in development.

**Returns**

void

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

Amount of logging in any 60 minute period is limited. See N/log Module Guidelines.

**Module**

N/log Module
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>string</td>
<td>Required</td>
<td>String to appear in the Title column on the Execution Log tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (&quot;&quot;'), or omit it, the word “Untitled” appears for the log entry.</td>
</tr>
<tr>
<td>details</td>
<td>any</td>
<td>Optional</td>
<td>You can pass any value for this parameter. If the value is a JavaScript object type, JSON.stringify(obj) is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.</td>
</tr>
</tbody>
</table>

Syntax

The following code snippet shows the syntax for this method.

```javascript
//Add additional code
...
var var1 = 'value';
log.debug({
    title: 'Debug Entry',
    details: 'Value of var1 is: ' + var1
});
...
//Add additional code
```

log.emergency(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Logs an entry of type EMERGENCY to the Execution Log tab of the script deployment for the current script. Use this method for scripts in production.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>Amount of logging in any 60 minute period is limited. See N/log Module Guidelines.</td>
</tr>
<tr>
<td>Module</td>
<td>N/log Module</td>
</tr>
</tbody>
</table>
Since 2016.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>string</td>
<td>Optional</td>
<td>String to appear in the Title column on the Execution Log tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (&quot;&quot;), or omit it, the word &quot;Untitled&quot; appears for the log entry.</td>
</tr>
<tr>
<td>details</td>
<td>any</td>
<td>Required</td>
<td>You can pass any value for this parameter. If the value is a JavaScript Object type, JSON.stringify(obj) is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.</td>
</tr>
</tbody>
</table>

Syntax

The following code snippet shows the syntax for this method.

```javascript
//Add additional code
...
var var1 = 'value';
log.emergency({
  title: 'Emergency Entry',
  details: 'Value of var1 is: ' + var1
});
...
//Add additional code
```

log.error(options)

Method Description

Logs an entry of type ERROR to the Execution Log tab of the script deployment for the current script. This entry will not appear on the Execution Log tab if the Log Level field for the script deployment is set to EMERGENCY or above. Use this method for scripts in production.

Returns

void

Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

Amount of logging in any 60 minute period is limited. See N/log Module Guidelines.
N/log Module

<table>
<thead>
<tr>
<th>Module</th>
<th>N/log Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
<td>string</td>
<td>Optional</td>
<td>String to appear in the <strong>Title</strong> column on the <strong>Execution Log</strong> tab of the script deployment. Maximum length is 99 characters. If you set this value to null, an empty string (''), or omit it, the word “Untitled” appears for the log entry.</td>
<td>2016.1</td>
</tr>
<tr>
<td>details</td>
<td>any</td>
<td>Required</td>
<td>You can pass any value for this parameter. If the value is a JavaScript object type, <code>JSON.stringify(obj)</code> is called on the object before displaying the value. NetSuite truncates any resulting string over 3999 characters.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

The following code snippet shows the syntax for this method.

```javascript
//Add additional code
...
var var1 = 'value';
log.error(
    title: 'Error Entry',
    details: 'Value of var1 is: ' + var1
);
...
//Add additional code
```

N/plugin Module

Load the N/plugin module to load custom plug-in implementations. For additional information, see the help topic **Custom Plug-ins**.

**Important:** You cannot use the SuiteScript Debugger to debug a script on demand that uses the N/plugin module. You must use deployed debugging. To use deployed debugging, you must complete the steps described in **Adding a Script that Instantiates a Custom Plug-in to NetSuite**. For the complete process on creating a custom plugin, see the help topic **Custom Plug-in Development**. For additional information about on demand and deployed debugging, see the help topic **Using the SuiteScript Debugger**.

- N/plugin Module Members
N/plugin Module Script Samples

N/plugin Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>plugin.##find##Implementations##(##options)</td>
<td>string[]</td>
<td>Server-side scripts</td>
<td>Returns the script IDs of custom plug-in type implementations.</td>
</tr>
<tr>
<td>Method</td>
<td>plugin.##load##Implementation##(##options)</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Instantiates an implementation of the custom plug-in type.</td>
</tr>
</tbody>
</table>

N/plugin Module Script Samples

To test the following samples, you need a custom plugin type with Script ID: customscript_magic_plugin and an interface with a single method: int doTheMagic(int, int).

⚠️ Important: You cannot use the SuiteScript Debugger to debug a script on demand that uses the N/plugin module. You must use deployed debugging. To use deployed debugging, you must complete the steps described in Adding a Script that Instantiates a Custom Plug-in to NetSuite. For the complete process on creating a custom plugin, see the help topic Custom Plug-in Development. For additional information about on demand and deployed debugging, see the help topic Using the SuiteScript Debugger.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

The following example shows an implementation of the interface:

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType plugintypeimpl
 */
define(function() {
  return {
    doTheMagic: function (operand1, operand2)
    {
      return operand1 + operand2;
    }
  }
});
```

The following Suitelet example iterates through all implementations of the custom plugin type customscript_magic_plugin.

⚠️ Important: The Suitelet script record must specify the plugin type under Custom Plug-in Types in order for it to recognize the plug-in.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
```
```javascript
define(['N/plugin'],
    function(plugin) {
        function onRequest(context) {
            var impls = plugin.findImplementations({
                type: 'customscript_magic_plugin'
            });
            for (i = 0; i < impls.length; i++) {
                var pl = plugin.loadImplementation({
                    type: 'customscript_magic_plugin',
                    implementation: impls[i]
                });
                log.debug('impl ' + impls[i] + ' result = ' + pl.doTheMagic(10, 20));
            }
            var pl = plugin.loadImplementation({
                type: 'customscript_magic_plugin'
            });
            log.debug('default impl result = ' + pl.doTheMagic(10, 20));
        }
        return {
            onRequest: onRequest
        };
    });

plugin.findImplementations(options)
```

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the script IDs of custom plug-in type implementations.</td>
<td>Returns an empty list when there is no custom plug-in type with the script ID available for the executing script.</td>
<td>Server-side scripts</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

### 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>options.type</td>
<td>string</td>
<td>required</td>
<td>The script ID of the custom plug-in type.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.includeDefault</td>
<td>boolean</td>
<td>optional</td>
<td>The default value is true, indicating that the default implementation should be included in the list.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.
plugin.loadImplementation(options)

**Method Description**

Instantiates an implementation of the custom plugin type.

Returns the implementation which is currently selected in the UI (Manage Plug-ins page) when no implementation ID is explicitly provided.

**Returns**

An Object implementing the custom plug-in type.

**Supported Script Types**

Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

**Module**

N/config Module

**Since**

2016.1

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The script ID of the custom plug-in type.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.implementation</td>
<td>string</td>
<td>optional</td>
<td>The script ID of the custom plug-in implementation.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Error Code**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNABLE_TO_FIND_IMPLEMENTATION_1_FOR_PLUGIN_2</td>
<td>Either there is no such implementation of the provided plug-in type, or the plug-in type does not exist.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/plugin Module Script Samples](#).

```
//Add additional code
...
var impls = plugin.findImplementations({
  type: 'customscript_sample_plugin'
});
...
//Add additional code
```
N/plugin Module

364

SuiteScript 2.0 API Reference

var pl = plugin.loadImplementation({
    type: 'customscript_sample_plugin' });
...
//Add additional code

N/portlet Module

Load the portlet module to resize or refresh a form portlet. See the help topic SuiteScript 2.0 Portlet Script Type.

N/portlet Module Members

N/portlet Module Script Sample

N/portlet Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>portlet.resize</td>
<td>void</td>
<td>Client scripts</td>
<td>Resizes a form portlet immediately.</td>
</tr>
<tr>
<td></td>
<td>portlet.refresh</td>
<td>void</td>
<td>Client scripts</td>
<td>Refreshes a form portlet immediately.</td>
</tr>
</tbody>
</table>

N/portlet Module Script Sample

The following sample shows how to create a form portlet that allows users to adjust its height and width. It creates two text fields representing the height and width of the portlet, measured in pixels. It also creates a button that runs the resize function to adjust the height and width of the portlet based on the values of the text fields.

The sample also shows how to create a button that uses the refresh function. When pressed, the portlet is updated to show the current date.

For more information about how a portlet is displayed on the NetSuite dashboard, see the help topic SuiteScript 2.0 Portlet Script Type.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType portlet
 * @NScriptPortletType form
 */
define([],
    function() {
        function render(context) {
            var portletObj = context.portlet;
            portletObj.title = 'Test Form Portlet';
            setComponentsForResize();
            setComponentsForRefresh();
            function setComponentsForResize() {
                var DEFAULT_HEIGHT = '50';
                var DEFAULT_WIDTH = '50';
                var inlineHTMLField = portletObj.addField({
```
N/portlet Module

SuiteScript 2.0 API Reference

id: 'divfield',
type: 'inlinehtml',
label: 'Test inline HTML'
});
inlineHTMLField.defaultValue = '<div id="divfield_elem" style="border: 1px dotted red; height: " + DEFAULT_HEIGHT + 'px; width: " + DEFAULT_WIDTH + 'px"></div>'
inlineHTMLField.updateLayoutType(
  layoutType: 'normal'
);
inlineHTMLField.updateBreakType(
  breakType: 'startcol'
);
var resizeHeight = portletObj.addField(
  id: 'resize_height',
type: 'text',
label: 'Resize Height'
));
resizeHeight.defaultValue = DEFAULT_HEIGHT;
var resizeWidth = portletObj.addField(
  id: 'resize_width',
type: 'text',
label: 'Resize Width'
));
resizeWidth.defaultValue = DEFAULT_WIDTH;
var resizeLink = portletObj.addField(
  id: 'resize_link',
type: 'inlinehtml',
label: 'Resize link'
));
resizeLink.defaultValue = resizeLink.defaultValue = "<a id='resize_link' onclick='require(['SuiteScripts/portletApiTestHelper'], function(portletApiTestHelper) {portletApiTestHelper.changeSizeAndResizePortlet(); })' href='#'>Resize</a><br>
";
}

function setComponentsForRefresh() {
  var textField = portletObj.addField(
    id: 'refresh_output',
type: 'text',
label: 'Date.now().toString()'
));
textField.defaultValue = Date.now().toString();
var refreshLink = portletObj.addField(
  id: 'refresh_link',
type: 'inlinehtml',
label: 'Refresh link'
));
refreshLink.defaultValue = "<a id='refresh_link' onclick='require(['SuiteScripts/portletApiTestHelper'], function(portletApiTestHelper) {portletApiTestHelper.refreshPortlet(); })' href='#'>Refresh</a>";
}

return {
  render: render
};
}

// portletApiTestHelper.js
```javascript
define(['N/portlet'],
    function(portlet) {
        function refreshPortlet() {
            portlet.refresh();
        }
        function resizePortlet() {
            var div = document.getElementById('divfield_elem');
            var newHeight = parseInt(document.getElementById('resize_height').value);
            var newWidth = parseInt(document.getElementById('resize_width').value);
            div.style.height = newHeight + 'px';
            div.style.width = newWidth + 'px';
            portlet.resize();
        }
        return {
            refreshPortlet: refreshPortlet,
            resizePortlet: resizePortlet
        };
    });
)
```

**portlet.resize**

- **Method Description**
  Resizes a form portlet type immediately.

- **Returns**
  Void

- **Supported Script Types**
  Client scripts

  For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

- **Governance**
  None

- **Module**
  N/portlet Module

- **Since**
  2016.1

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/portlet Module Script Sample](#).

```javascript
//Add additional code
...
portlet.resize();
...
//Add additional code
```

**portlet.refresh**

- **Method Description**
  Refreshes a form portlet type immediately.

- **Returns**
  Void

- **Supported Script Types**
  Client scripts

  For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).
N/portlet Module

Governance
None

Module
N/portlet Module

Since
2016.1

Syntax

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/portlet Module Script Sample.

```javascript
... 
portlet.refresh();
... 
```

N/query Module

Load the query module to create and run queries using the SuiteAnalytics Workbook query engine. For more information, see the help topic SuiteAnalytics Workbook. Using the query module, you can:

- Use multilevel joins to create queries using field data from multiple record types.
- Create conditions (filters) using AND, OR, and NOT logic, as well as formulas and relative dates.
- Sort query results based on the values of multiple columns.
- Load and delete existing saved queries that were created using the SuiteAnalytics Workbook interface.
- View paged query results.
- Use promises for asynchronous execution.

For more information about creating scripts using the N/query module, see the following help topics:

- Scripting with the N/query Module
- Formulas in the N/query Module
- Relative Dates in the N/query Module

**Important:** As you use the N/query module, keep the following considerations in mind:

- The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. In the 2019.1 release, you can use the N/query module to load and delete existing queries, but you cannot save queries. You can save queries using the SuiteAnalytics Workbook interface. For more information, see the help topic Navigating SuiteAnalytics Workbook.
- The N/query module supports the same record types that are supported in the SuiteAnalytics Workbook interface. For more information, see the help topic Available Records.

- N/query Module Members
- Column Object Members
- Component Object Members
- Condition Object Members
- Page Object Members
- PagedData Object Members
- PageRange Object Members
## N/query Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>query.Column</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the field types (query result columns) that are displayed from the query results. Use <code>Query.createColumn(options)</code> or <code>Component.createColumn(options)</code> to create this object.</td>
</tr>
<tr>
<td></td>
<td>query.Component</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates one component of the query definition. The query definition always contains at least one component that encapsulates the initial search type. Queries with joins contain multiple components that encapsulate the join relationships. The initial component (<code>query.root</code>) is automatically created with the query definition (<code>query.Query</code>). Use <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code> to create subsequent components.</td>
</tr>
<tr>
<td></td>
<td>query.Condition</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a condition. A condition narrows the query results. Use <code>Query.createCondition(options)</code> or <code>Component.createCondition(options)</code> to create this object.</td>
</tr>
<tr>
<td></td>
<td>query.Page</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates one page of the paged query results.</td>
</tr>
<tr>
<td></td>
<td>query.PagedData</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a set of paged query results. This object also contains information about the set of paged results it encapsulates.</td>
</tr>
<tr>
<td></td>
<td>query.PageRange</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a range of pages from the paged query results.</td>
</tr>
<tr>
<td></td>
<td>query.RelativeDate</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a relative date range to use in query conditions.</td>
</tr>
<tr>
<td></td>
<td>query.Result</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a single row of the query result set.</td>
</tr>
<tr>
<td></td>
<td>query.ResultSet</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the set of results returned by the query.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>query.Query</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the query definition. Use query.create(options) or query.load(options) to create this object. The creation of this object is the first step in creating a query with the N/query Module.</td>
</tr>
<tr>
<td></td>
<td>query.Sort</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a sort that is placed on a particular query result column. Use Query.createSort(options) or Component.createSort(options) to create this object.</td>
</tr>
<tr>
<td>Method</td>
<td>query.##create(options)</td>
<td>query.Query</td>
<td>Client and server-side scripts</td>
<td>Creates the query definition. The execution of this method is the first step in creating a query with the N/query Module.</td>
</tr>
<tr>
<td></td>
<td>query.##create##Date##Range##(##options)</td>
<td>query.##RelativeDate</td>
<td>Client and server-side scripts</td>
<td>Creates a query.##RelativeDate object that represents a range of dates relative to the current date.</td>
</tr>
<tr>
<td></td>
<td>query.##delete(options)</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Deletes an existing query that was created using the SuiteAnalytics Workbook UI. The deleted query is no longer available and cannot be modified or executed.</td>
</tr>
<tr>
<td></td>
<td>query.load(options)</td>
<td>query.Query</td>
<td>Client and server-side scripts</td>
<td>Loads an existing query that was created using the SuiteAnalytics Workbook UI. The loaded query can be modified (for example, by setting additional property values), joined with other search types, and executed in the same way as queries created using query.create(options).</td>
</tr>
<tr>
<td>Enum</td>
<td>query.Aggregate</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Holds the string values for aggregate functions supported with the ## N/query Module. This enum is used to pass the aggregate function argument to ## Component.##createColumn(options),## Component.##createCondition(options),## Query.##createColumn(options), and ## Query.##createCondition(options).</td>
</tr>
<tr>
<td></td>
<td>query.DateRangeId</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Holds the string values for supported date range codes in relative dates. This enum is used to pass the date range argument to query.createDateRange(options).</td>
</tr>
<tr>
<td></td>
<td>query.Operator</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Holds the string values for operators supported with the ## N/query Module. This enum is used to pass the operator argument to ## Query.##createCondition(options) ## and ## Component.##createCondition(options).</td>
</tr>
</tbody>
</table>
### N/query Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>query.RelativeDateRange</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Holds <code>query.RelativeDate</code> object values for supported date ranges in relative dates. This enum is used to pass the values argument to <code>Query.createCondition(options)</code> and <code>Component.createCondition(options)</code>.</td>
</tr>
<tr>
<td></td>
<td>query.ReturnType</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Holds the string values for the formula return types supported with the <code>N/query Module</code>. This enum is used to pass the formula return type argument to <code>Query.createColumn(options)</code>, <code>Component.createColumn(options)</code>, <code>Query.createCondition(options)</code>, and <code>Component.createCondition(options)</code>.</td>
</tr>
<tr>
<td></td>
<td>query.SortLocale</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Holds the string values for sort locales supported with the <code>N/query Module</code>. This enum is used to pass the sort locale argument to <code>Query.createSort(options)</code> and <code>Component.createSort(options)</code>.</td>
</tr>
<tr>
<td></td>
<td>query.Type</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Holds the string values for supported search types used in the query definition. This enum is used to pass the initial search type argument to <code>query.create(options)</code>.</td>
</tr>
</tbody>
</table>

### Column Object Members

The following members are called on the `query.Column` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Column.aggregate</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes an aggregate function that is performed on the query result column. An aggregate function performs a calculation on the column values and returns a single value.</td>
</tr>
<tr>
<td></td>
<td>Column.component</td>
<td>query.Component (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds a reference to the <code>query.Component</code> object to which this query result column belongs.</td>
</tr>
<tr>
<td></td>
<td>Column.consolidated</td>
<td>boolean (read-only)</td>
<td>Client and server-side scripts</td>
<td>Indicates whether the query result column shows consolidated currency amounts.</td>
</tr>
<tr>
<td></td>
<td>Column.fieldId</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds the name of the query result column. This property and the <code>Column.formula</code> property cannot be set at the same time.</td>
</tr>
</tbody>
</table>
### Column Object Members

**Member**<br>**Type**<br>**Name**<br>**Return Type/Value Type**<br>**Supported Script Types**<br>**Description**

| Column.formula | string (read-only) | Client and server-side scripts | Describes the formula used to create the query result column. This property and the Column.fieldId property cannot be set at the same time. |
| Column.groupBy | boolean (read-only) | Client and server-side scripts | Indicates whether the query results are grouped by this query result column. |
| Column.type | string (read-only) | Client and server-side scripts | Describes the return type of the formula used to create the query result column. |

### Component Object Members

The following members are called on the `query.Component` object.

**Member Type**<br>**Name**<br>**Return Type/Value Type**<br>**Supported Script Types**<br>**Description**

<p>| Method | Component.###autoJoin###(options) | query.###Component### | Client and server-side scripts | Creates a join relationship. After you create the initial query definition, use Query.autoJoin(options) to create your first join. Then use this method to create each subsequent join. |
| Method | Component.###createColumn###(options) | query.Column | Client and server-side scripts | Creates a query result column based on the component. Use this method to create columns based on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). |
| Method | Component.###createCondition###(options) | query.###Condition### | Client and server-side scripts | Creates a condition (filter column) based on the component. Use this method to create conditions based on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). |
| Method | Component.###createSort###(options) | query.Sort | Client and server-side scripts | Creates a sort based on the component. Use this method to create sorts based on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options). |
| Method | Component.###join###(options) | query.###Component### | Client and server-side scripts | Creates a join relationship. This method is an alias to Component.autoJoin(options). After you create the initial query definition, use Query.autoJoin(options) to create your first join. Then use this method, or Component.autoJoin(options), to create each subsequent join. |
| Method | Component.###joinFrom###(options) | query.###Component### | Client and server-side scripts | Creates an explicit directional join relationship from another component |</p>
<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Component.source</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the source search type of the component. The value of this property is set when Component.joinFrom(options) is called to perform an explicit directional join from another component.</td>
</tr>
<tr>
<td>Property</td>
<td>Component.target</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the target search type of the component. The value of this property is set when Component.joinTo(options) is called to perform an explicit directional join to another component.</td>
</tr>
<tr>
<td>Property</td>
<td>Component.type</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the search type of the component.</td>
</tr>
<tr>
<td>Property</td>
<td>Component.child</td>
<td>Object (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the child components of the component. This property holds an object of key/value pairs. Each key is the name of a child component. Each value is the corresponding child query.Component object.</td>
</tr>
<tr>
<td>Property</td>
<td>Component.parent</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the parent query.Component object of the component.</td>
</tr>
</tbody>
</table>

**Condition Object Members**

The following members are called on the query.Condition object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Condition.aggregate</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes an aggregate function that is performed on the condition. An aggregate function</td>
</tr>
</tbody>
</table>
### N/query Module

#### SuiteScript 2.0 API Reference

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Member</td>
<td></td>
<td></td>
<td></td>
<td>perform a calculation on the condition values and returns a single value.</td>
</tr>
<tr>
<td>Condition</td>
<td>children</td>
<td>query.Condition[]</td>
<td>Client and server-side scripts</td>
<td>Holds an array of child conditions used to create the parent condition.</td>
</tr>
<tr>
<td>Component</td>
<td>component</td>
<td>query.Component</td>
<td>Client and server-side scripts</td>
<td>Holds a reference to the query.Component object to which this condition belongs.</td>
</tr>
<tr>
<td>Condition</td>
<td>fieldId</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds the name of the condition.</td>
</tr>
<tr>
<td>Condition</td>
<td>formula</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the formula used to create the condition.</td>
</tr>
<tr>
<td>Condition</td>
<td>operator</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds the name of the operator used to create the condition.</td>
</tr>
<tr>
<td>Condition</td>
<td>type</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>The return type of the formula used to create the condition.</td>
</tr>
<tr>
<td>Condition</td>
<td>values</td>
<td>string[] (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds an array of values used by an operator to create the condition.</td>
</tr>
</tbody>
</table>

### Page Object Members

The following members are called on the query.Page object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Page.data</td>
<td>query.ResultSet (read-only)</td>
<td>Client and server-side scripts</td>
<td>References the query results contained in this page.</td>
</tr>
<tr>
<td></td>
<td>Page.isFirst</td>
<td>boolean (read-only)</td>
<td>Client and server-side scripts</td>
<td>Indicates whether this page is the first of the paged query results.</td>
</tr>
<tr>
<td></td>
<td>Page.isLast</td>
<td>boolean (read-only)</td>
<td>Client and server-side scripts</td>
<td>Indicates whether this page is the last of the paged query results.</td>
</tr>
<tr>
<td></td>
<td>Page.pagedData</td>
<td>query.PagedData (read-only)</td>
<td>Client and server-side scripts</td>
<td>References the set of paged query results that this page is from.</td>
</tr>
<tr>
<td></td>
<td>Page.pageRange</td>
<td>query.PageRange (read-only)</td>
<td>Client and server-side scripts</td>
<td>The range of query results for this page.</td>
</tr>
</tbody>
</table>
## PagedData Object Members

The following members are called on the `query.PagedData` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>PagedData.iterator()</td>
<td>Iterator object</td>
<td>Client and server-side scripts</td>
<td>Standard SuiteScript 2.0 object for iterating through results.</td>
</tr>
<tr>
<td>Property</td>
<td>PagedData.count</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the total number of paged query results.</td>
</tr>
<tr>
<td></td>
<td>PagedData.pageRanges</td>
<td>query.PageRange[]</td>
<td>Client and server-side scripts</td>
<td>Holds an array of page ranges for the set of paged query results.</td>
</tr>
<tr>
<td></td>
<td>PagedData.pageSize</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the number of query result rows per page.</td>
</tr>
</tbody>
</table>

## PageRange Object Members

The following members are called on the `query.PageRange` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>PageRange.index</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the array index for this page range.</td>
</tr>
<tr>
<td></td>
<td>PageRange.size</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the number of query result rows in this page range.</td>
</tr>
</tbody>
</table>

## Query Object Members

The following members are called on the `query.Query` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Query.and()</td>
<td>query.Condition[]</td>
<td>Client and server-side scripts</td>
<td>Creates a new condition (a <code>query.Condition</code> object) that corresponds to a logical conjunction (AND) of the arguments passed to the method. The arguments must be one or more <code>query.Condition</code> objects.</td>
</tr>
<tr>
<td></td>
<td>Query.autoJoin()</td>
<td>query.Component[]</td>
<td>Client and server-side scripts</td>
<td>Creates a join relationship.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>After you create the initial query definition, use this method to create your first join.</td>
</tr>
<tr>
<td></td>
<td>Query.createColumn()</td>
<td>query.Column</td>
<td>Client and server-side scripts</td>
<td>Creates a query result column based on the <code>query.Query</code> object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Use this method to create columns on the initial query definition created with <code>query.create(options)</code>.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type/Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------------------------</td>
<td>------------------------</td>
<td>------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>createCondition</strong></td>
<td>query.<strong>Condition</strong></td>
<td>Client and server-side scripts</td>
<td>Creates a condition (filter column) based on the query.<strong>Query</strong> object. Use this method to create conditions on the initial query definition created with query.<strong>create</strong>(options).</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>createSort</strong></td>
<td>query.<strong>Sort</strong></td>
<td>Client and server-side scripts</td>
<td>Creates a sort based on the query.<strong>Query</strong> object. The query.<strong>Sort</strong> object describes a sort that is placed on a particular query result column or condition.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>join</strong>(options)</td>
<td>query.<strong>Component</strong></td>
<td>Client and server-side scripts</td>
<td>Creates a join relationship. This method is an alias to Query.<strong>autoJoin</strong>(options). After you create the initial query definition, use this method, or Query.<strong>autoJoin</strong>(options), to create your first join.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>joinFrom</strong>(options)</td>
<td>query.<strong>Component</strong></td>
<td>Client and server-side scripts</td>
<td>Creates an explicit directional join relationship from another component to the root component of the search definition (an inverse join). This method sets the Component.<strong>source</strong> property on the returned query.<strong>Component</strong> object. After you create the initial query definition, use this method to create your first join as an explicit directional join from another component to this component.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>joinTo</strong>(options)</td>
<td>query.<strong>Component</strong></td>
<td>Client and server-side scripts</td>
<td>Creates an explicit directional join relationship to another component from this component (a polymorphic join). You can use this method to specify the target of the join when a field can join multiple search types. This method sets the Component.<strong>target</strong> property on the returned query.<strong>Component</strong> object. After you create the initial query definition, use this method to create your first join as an explicit directional join to another component from this component.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>not</strong></td>
<td>query.<strong>Condition</strong></td>
<td>Client and server-side scripts</td>
<td>Creates a new condition (a query.<strong>Condition</strong> object) that corresponds to a logical negation (NOT) of the argument passed to the method. The argument must be a query.<strong>Condition</strong> object.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>or</strong></td>
<td>query.<strong>Condition</strong></td>
<td>Client and server-side scripts</td>
<td>Creates a new condition (a query.<strong>Condition</strong> object) that corresponds to a logical disjunction (OR) of the arguments passed to the method. The arguments must be one or more query.<strong>Condition</strong> objects.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>run</strong></td>
<td>query.<strong>ResultSet</strong></td>
<td>Client and server-side scripts</td>
<td>Executes the query and returns the query result set.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>run.promise</strong></td>
<td>query.<strong>ResultSet</strong></td>
<td>Client scripts</td>
<td>Executes the query asynchronously and returns the query result set.</td>
</tr>
<tr>
<td></td>
<td>Query.<strong>runPaged</strong></td>
<td>query.<strong>PagedData</strong></td>
<td>Client and server-side scripts</td>
<td>Executes the query and returns a set of paged results.</td>
</tr>
</tbody>
</table>
## Member Type

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>query.runPaged</td>
<td>query.PagedData</td>
<td>Client scripts</td>
<td>Executes the query asynchronously and returns a set of paged results.</td>
</tr>
</tbody>
</table>

### Property

- **Query.child**
  - **Object (read-only)**
  - **Client and server-side scripts**
  - **Holds a reference to children of the root component of the query definition. The value of this property is an object of key/value pairs. Each key is the name of a child component. Each respective value is the corresponding query_Component object.**

- **Query.columns**
  - **query.Column[]}**
  - **Client and server-side scripts**
  - **Holds an array of query result columns returned from the query. Before you execute the query, you must assign all created columns as array values to this property.**

- **Query.condition**
  - **query.Condition}}**
  - **Client and server-side scripts**
  - **References the parent condition that narrows the query results. Before you execute the query, you must assign your simple or complex conditions to this property.**

- **Query.id**
  - **number (read-only)**
  - **Client and server-side scripts**
  - **Holds the ID of the query definition. This property has a value only for existing queries that are loaded using query.load(options). If you create a query using query.create(options) but do not save it, this property is null.**

- **Query.name**
  - **string (read-only)**
  - **Client and server-side scripts**
  - **Holds the name of the query definition. This property has a value only for existing queries that are loaded using query.load(options). If you create a query using query.create(options) but do not save it, this property is null.**

- **Query.root**
  - **query.Component}}**
  - **Client and server-side scripts**
  - **References the root component of the query definition.**

- **Query.sort**
  - **query.Column[]}**
  - **Client and server-side scripts**
  - **Holds an array of query result columns used for sorting.**

- **Query.type**
  - **string (read-only)**
  - **Client and server-side scripts**
  - **Holds the search type of the initial query definition.**

### RelativeDate Object Members

The following members are called on the `query.RelativeDate` object.

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RelativeDate.dateRangeId</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds the ID of the relative date range. Represents the end point of the relative date range.</td>
</tr>
</tbody>
</table>
### Result Object Members

The following members are called on the `query.Result` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Result.columns</td>
<td>query.Column[] (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds an array of query result column references.</td>
</tr>
<tr>
<td></td>
<td>Result.values</td>
<td>Array&lt;string</td>
<td>number</td>
<td>boolean</td>
</tr>
</tbody>
</table>

### ResultSet Object Members

The following members are called on the `query.ResultSet` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>ResultSet.iterator()</td>
<td>Iterator object</td>
<td>Client and server-side scripts</td>
<td>Standard SuiteScript 2.0 object for iterating through results.</td>
</tr>
<tr>
<td>Property</td>
<td>ResultSet.columns</td>
<td>query.Column[] (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds an array of query result column references.</td>
</tr>
<tr>
<td></td>
<td>ResultSet.results</td>
<td>query.Result[] (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds an array of <code>query.Result</code> objects.</td>
</tr>
<tr>
<td></td>
<td>ResultSet.types</td>
<td>string[] (read-only)</td>
<td>Client and server-side scripts</td>
<td>Holds an array of the return types for <code>ResultSet.results</code>.</td>
</tr>
</tbody>
</table>

### Sort Object Members

The following members are called on the `query.Sort` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Sort.ascending</td>
<td>boolean</td>
<td>Client and server-side scripts</td>
<td>Indicates whether the sort direction is ascending.</td>
</tr>
</tbody>
</table>
### N/query Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type/Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sort.caseSensitive</td>
<td>boolean</td>
<td>Client and server-side scripts</td>
<td>Indicates whether the sort is case sensitive. If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same.</td>
</tr>
<tr>
<td></td>
<td>Sort.column</td>
<td>query.Column (read-only)</td>
<td>Client and server-side scripts</td>
<td>Describes the query result column that the query results are sorted by.</td>
</tr>
<tr>
<td></td>
<td>Sort.locale</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The locale to use for the sort. A locale represents a combination of language and region, and it can affect how certain values (such as strings) are sorted.</td>
</tr>
<tr>
<td></td>
<td>Sort.nullsLast</td>
<td>boolean</td>
<td>Client and server-side scripts</td>
<td>Indicates whether query results with null values are listed at the end of the query results.</td>
</tr>
</tbody>
</table>

### N/query Module Script Samples

**Note:** These sample scripts use the `require` function so that you can copy them into the debugger and test them. Keep in mind that you must use the `define` function in your entry point script (the script that you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

The following script creates a query for Customer records, joins the query with two other query types, and runs the query:

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/query'],
function(query) {

  // Create a query definition for Customer records
  var search = query.create({
    type: query.Type.CUSTOMER
  });

  // Join the original query definition based on the salesrep field. In a Customer
  // record, the salesrep field contains a reference to an Employee record. When you
  // join based on this field, you are joining the query definition with the Employee
  // query type, and you can access the fields of the joined Employee record in
  // your query.
  var salesrep = search.autoJoin({
```
fieldId: 'salesrep'
});

// Join the joined query definition based on the location field. In an Employee
// record, the location field contains a reference to a Location record.
var location = salesrep.autoJoin({
    fieldId: 'location'
});

// Create conditions for the query
var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

// Combine conditions using and() and or() operator methods. In this example,
// the combined condition states that the id field of the Customer record must
// have a value of either 107 or 2647, and the email field of the Employee
// record (the record that is referenced in the salesrep field of the Customer
// record) must not start with 'foo'.
search.condition = search.and(cond3, search.or(cond1, cond2));

// Create query columns
search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    }),
    location.createColumn({
        fieldId: 'name'
    })]
The following script creates a query for Transaction records, joins the query with another query type, and runs the query as a paged query:

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/query'],
function(query) {

    // Create a query definition for Transaction records
    var search = query.create({
        type: query.Type.TRANSACTION
    });

    // Join the original query definition based on the employee field. In a Transaction
    // record, the employee field contains a reference to an Employee record. When you
    // join based on this field, you are joining the query definition with the Employee
    // query type, and you can access the fields of the joined Employee record in
    // your query.
    var entity = search.autoJoin({
        fieldId: 'employee'
    });

    // Create a query column
    search.columns = [
        entity.createColumn({
            fieldId: 'subsidiary'
        })
    ];

    // Sort the query results based on a query column
    search.sort = [
        search.createSort({
            column: search.columns[3]
        }),
        search.createSort({
            column: search.columns[0],
            ascending: false
        })
    ];

    // Run the query
    var resultSet = search.run();

    // Retrieve and log the results
    var results = resultSet.results;
    for (var i = results.length - 1; i >= 0; i--)
        log.debug(results[i].values);
    log.debug(resultSet.types);
});
```
search.createSort({
    column: search.columns[0],
    ascending: false
});

// Run the query as a paged query with 10 results per page
var results = search.runPaged({
    pageSize: 10
});

log.debug(results.pageRanges.length);
log.debug(results.count);

// Retrieve the query results using an iterator
var iterator = results.iterator();
iterator.each(function(result) {
    var page = result.value;
    log.debug(page.pageRange.size);
    return true;
});

// Alternatively, retrieve the query results by looping through
// each result
for (var i = 0; i < results.pageRanges.length; i++) {
    var page = results.fetch(i);
    log.debug(page.pageRange.size);
}

Scripting with the N/query Module

The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. Before you start creating your queries, you should be familiar with the module objects and how to use them, as well as some of the terminology used in the N/query module. You can also take a look at a script walkthrough that explains how to create queries using different approaches.

- N/query Module Objects
- N/query Module Terminology

N/query Module Objects

The N/query module includes the following objects:

- Query and Component Objects
- Condition Object
- RelativeDate Object
- Column Object
- Sort Object
- ResultSet and Result Objects
- Page, PagedData, and PageRange Objects
Query and Component Objects

The `query.Query` object and the `query.Component` object are the primary building blocks for a query created with the N/query module. Each query creates one `query.Query` object and one or more `query.Component` objects. The `query.Query` object encapsulates the query definition, and the `query.Component` object encapsulates one component of the query definition.

To create a query with the N/query module:

1. Use the `query.create(options)` method to create your initial query definition (the `query.Query` object). The initial query definition uses one search type. For available search types, see `query.Type`.
2. After you create the initial query definition, use `Query.autoJoin(options)`, `Query.joinFrom(options)`, or `Query.joinTo(options)` to create your first join.
3. Use `Component.autoJoin(options)`, `Component.joinFrom(options)`, or `Component.joinTo(options)` to create all subsequent joins.

The query definition always contains at least one `query.Component` object. Each new component is created as a child of the previous component, and all components exist as children of the query definition. You can think of a component as a building block; each new component builds on the previous component created. The last component created encapsulates the relationship between it and all of its parent components.

Queries with joins contain multiple components. The query definition contains a child `query.Component` object for each of the following:

- **The initial query definition**: The initial `query.Component` object is called the root component. It encapsulates the initial search type passed to `query.create(options)`. The root component is automatically created with the initial query definition and is a child to the `query.Query` object. The `Query.root` property contains a reference to the root component.
- **The first join**: The second `query.Component` object is created with `Query.autoJoin(options)`, `Query.joinFrom(options)`, or `Query.joinTo(options)`. It encapsulates the relationship between the initial query definition and the second search type. This relationship is determined by the join ID passed to these methods, as well as whether `Query.joinFrom(options)` or `Query.joinTo(options)` was used to create an explicit directional join. The second `query.Component` object is a child to the root component.
- **Each subsequent join**: The third `query.Component` object is created with `Component.autoJoin(options)`, `Component.joinFrom(options)`, or `Component.joinTo(options)`. All subsequent joins are also created using these methods. Each of these `query.Component` objects encapsulates the relationship between all previous search types and the new search type. This relationship is determined by the join ID passed to these methods, as well as whether `Component.joinFrom(options)` or `Component.joinTo(options)` was used to create an explicit directional join.

Condition Object

A condition narrows the query results. The `query.Condition` object performs the same function as the `search.Filter` object in the N/search Module. The primary difference is that `query.Condition` objects can contain other `query.Condition` objects.

To create conditions:

- Use `Query.createCondition(options)` to create conditions for the initial query definition created with `query.create(options)`.
- Use `Component.createCondition(options)` to create conditions for the join relationships created with `Query.autoJoin(options)`, `Query.joinFrom(options)/Query.joinTo(options)`, `Component.autoJoin(options)`, or `Component.joinFrom(options)/Component.joinTo(options)`.
If you have multiple conditions, use `Query.and()`, `Query.or()`, and `Query.not()` to create a new nested condition.

If you want to use a formula to define your conditions, assign the formula to `Condition.formula`.

Assign your simple or nested conditions as array values to `Query.condition`.

**RelativeDate Object**

The `query.RelativeDate` object represents a date range that is relative to the current date. You can use relative dates when you create query conditions.

To create relative dates:

- Use `query.createDateRange(options)` to create a `query.RelativeDate` object. When you call `query.createDateRange(options)`, use the values in the `query.DateRangeId` enum to specify a range of dates relative to the current date.
- Use `Query.createCondition(options)` or `Component.createCondition(options)` to create a condition using the `query.RelativeDate` object. Alternatively, you can create a condition using values in the `query.RelativeDateRange` enum.
- If you have multiple conditions, use `Query.and()`, `Query.or()`, and `Query.not()` to create a new nested condition.
- Assign your simple or nested conditions as array values to `Query.condition`.

**Column Object**

The `query.Column` object is the equivalent of the `search.Column` object in the `N/search Module`. The `query.Column` object describes the field types (columns) that are displayed from the query results.

To create columns:

- Use `Query.createColumn(options)` to create a column on the initial query definition created with `query.create(options)`.
- Use `Component.createColumn(options)` to create a column on a join relationship created with `Query.autoJoin(options)`, `Query.joinFrom(options)/Query.joinTo(options)`, `Component.autoJoin(options)`, or `Component.joinFrom(options)/Component.joinTo(options)`.
- If you want to use a formula to define your columns, assign the formula to `Column.formula`.
- Assign all created columns as array values to `Query.columns`.

**Sort Object**

The `query.Sort` object describes how query results are sorted (for example, ascending or descending, case sensitive or case insensitive, and so on).

To create a sort:

- Use `Query.createSort(options)` to create a sort on the initial query definition created with `query.create(options)`.
- Use `Component.createSort(options)` to create a sort based on a join relationship created with `Query.autoJoin(options)`, `Query.joinFrom(options)/Query.joinTo(options)`, `Component.autoJoin(options)`, or `Component.joinFrom(options)/Component.joinTo(options)`.
- Assign all created sorts as array values to `Query.sort`.

**ResultSet and Result Objects**

When you are ready to execute your query, call `Query.run()`. This method returns a `query.ResultSet` object, which encapsulates the metadata for the set of results returned by the query.
To access your actual query results, iterate through the `ResultSet.results` array. Each member of the `ResultSet.results` array is a `query.Result` object. The `query.Result` object encapsulates a single row of the result set.

**Page, PagedData, and PageRange Objects**

You also can execute your query by calling `Query.runPaged()`. This method returns a `query.PagedData` object, which encapsulates a set of paged query results.

To access your query results, iterate through the paged query results using `PagedData.iterator()`. You can access each page of the query results, which are represented by `query.Page` objects. The `query.PageRange` object encapsulates the range of query results for a page.

### N/query Module Terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate function</td>
<td>An aggregate function performs a calculation on a column of values and returns a single value. You can add aggregate functions to conditions and query results columns.</td>
<td><code>query.Aggregate</code>&lt;br&gt;<code>Component.#createColumn(options)</code>&lt;br&gt;<code>Component.#createCondition##(options)</code>&lt;br&gt;<code>Query.#createColumn(options)</code>&lt;br&gt;<code>Query.#createCondition##(options)</code></td>
</tr>
<tr>
<td>Column</td>
<td>A column describes the field types (columns) that are displayed from the query results. A column is also known as a query results column.</td>
<td><code>query.Column</code></td>
</tr>
<tr>
<td>Component</td>
<td>When you script queries with the N/query module, your query is made up of one or more components, which are represented as <code>query.Component</code> objects. You can think of a component as a building block; each new component builds on the previous component created. &lt;br&gt;  - The first component created represents the initial search type and is a child of <code>query.Query</code>. &lt;br&gt;  - Each subsequent component created is a child of the previous component. &lt;br&gt;  - The last component created encapsulates the join relationship between it and all of its parent components. &lt;br&gt; A query always contains at least one component: the root component. When you create the initial query definition using <code>query.create(options)</code>, the root component is created automatically. Queries with joins contain multiple components. A new component is created each time you create a join using one of the following methods: &lt;br&gt;  - <code>Query.autoJoin(options)</code>, <code>Query.joinFrom(options)</code>, or <code>Query.joinTo(options)</code>  &lt;br&gt;  - <code>Component.autoJoin(options)</code>, <code>Component.joinFrom(options)</code>, or <code>Component.joinTo(options)</code></td>
<td><code>query.Component</code></td>
</tr>
<tr>
<td>Condition</td>
<td>A condition narrows the query results.</td>
<td><code>query.Condition</code></td>
</tr>
<tr>
<td>Formula</td>
<td>Formulas can be used to create conditions and columns.</td>
<td><code>Formulas in Search</code>&lt;br&gt;<code>SQL Expressions</code></td>
</tr>
<tr>
<td>Group</td>
<td>You can summarize your query results into unique groups of column values.</td>
<td><code>Column.groupBy</code></td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
<td>For More Information</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>Join</td>
<td>A join lets you create a query based on a field type that is shared between two record types. You can use <code>query.autoJoin(options)</code> and <code>Component.autoJoin(options)</code> to create a join relationship automatically based on a field that you specify. You can use <code>query.joinFrom(options)</code> and <code>Component.joinFrom(options)</code> and <code>query.joinTo(options)</code> and <code>Component.joinTo(options)</code> to create explicit directional join relationships from one component to another.</td>
<td>query.Query, query.Component</td>
</tr>
<tr>
<td>Page</td>
<td>A page represents one page from a set of paged query results. When you create a query with the N/query module, you can return the results as one result set or a set of paged results.</td>
<td>Query.runPaged(), query.PagedData, query.PageRange, query.Page</td>
</tr>
<tr>
<td>Paged data</td>
<td>Paged data represents a set of paged query results.</td>
<td>Query.runPaged(), query.PagedData, query.PageRange, query.Page</td>
</tr>
<tr>
<td>Page range</td>
<td>A page range is a set of pages from a set of paged query results.</td>
<td>Query.runPaged(), query.PagedData, query.PageRange, query.Page</td>
</tr>
<tr>
<td>Relative date</td>
<td>A relative date is a range of dates that is relative to the current date. You can use relative dates when you create query conditions.</td>
<td>query.RelativeDate, query.RelativeDateRange</td>
</tr>
<tr>
<td>Result</td>
<td>A result is a single row from a result set.</td>
<td>Query.run(), query.ResultSet, query.Result</td>
</tr>
<tr>
<td>Result set</td>
<td>A result set is a set of query results.</td>
<td>Query.run(), query.ResultSet, query.Result</td>
</tr>
<tr>
<td>Query definition</td>
<td>The query definition is the initial search type you define, plus any subsequent joins you define. The initial query definition is created with <code>query.create(options)</code>.</td>
<td>query.Query</td>
</tr>
<tr>
<td>Search type</td>
<td>The search type is the initial search type of your query definition. It represents the record type you want to search for. It is set with the <code>query.Type</code> enum during the execution of <code>query.create(options)</code>. For example, if you want to search for customer records, specify <code>query.Type.CUSTOMER</code> as the search type when you call <code>query.create(options)</code>.</td>
<td>query.Query, query.Type</td>
</tr>
<tr>
<td>Sort</td>
<td>A sort is placed on a query results column to describe how the query results are sorted (for example, ascending or descending, case sensitive or case insensitive, and so on).</td>
<td>query.Sort, Query.createSort(options), Component.createSort(options)</td>
</tr>
</tbody>
</table>
Formulas in the N/query Module

When you create a query using the N/query module, you can specify columns and conditions for the query. Columns describe the field types (or columns) that are displayed from the query results, and conditions narrow the query results based on certain criteria. You create a column using `Query.createColumn(options)`, and you create a condition using `Query.createCondition(options)`. Both of these methods let you create a column or condition in two ways:

- Use the `fieldId` parameter to explicitly specify the field on which to create the column or condition.
- Use the `formula` parameter to specify a formula to create the column or condition.

You can use formulas to perform a calculation to determine the column or condition value based on the values of other fields in the record. For example, consider a situation in which you are working with Customer records that include custom fields. These custom fields contain the amount of stock for various items (50 units of item A, 24 units of item B, and so on). In your query results, you want to include a column that calculates and displays the total amount of stock for all items for a Customer. If the Customer records include three custom stock fields, you can create the result column as follows:

```javascript
query.createColumn({
  formula: '{item_A_stock} + {item_B_stock} + {item_C_stock}',
  type: query.ReturnType.INTEGER
});
```

When you use a formula to create a column or condition, you must also use the `type` parameter to specify the return type of the formula. This parameter accepts values from the `query.ReturnType` enum. Defining the formula’s return type might be required if the return type cannot be determined automatically based on the formula. When you set the `type` parameter, the return value is properly formatted based on the data type that you specify.

For more information on formulas, see the help topics Formulas in Search and SQL Expressions.

Formulas in Joined Queries

You can join your queries with other record types. Joining queries lets you obtain and display query results with field values from multiple record types. When you use a formula in a joined query, you must use fully qualified field IDs to access the fields in each joined record type. You must specify the full join trail from the base record type. The join trail differs depending on the record types and join type.

Use the `^` and `<` operators to access fields in joined queries. The `^` operator lets you access fields in record types that are joined using `Query.joinTo(options)` or `Component.joinTo(options)`. This type of join is also known as a polymorphic join. The `<` operator lets you access fields in record types that are joined using `Query.joinFrom(options)` or `Component.joinFrom(options)`. This type of join is also known as an inverse join. You can use these operators when working with formulas in SuiteScript or the NetSuite UI.

The following table lists common join operations and the corresponding join trail.

<table>
<thead>
<tr>
<th>Join Type</th>
<th>Join Operation</th>
<th>Join Trail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic using <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code></td>
<td>// The base record type is Customer&lt;br&gt;var myCustomerQuery = query.create({&lt;br&gt;  type: query.Type.CUSTOMER&lt;br&gt;});</td>
<td>Base record fields (Customer)&lt;br&gt;■ customer.&lt;baseFieldName&gt;&lt;br&gt;■ Example: customer.email&lt;br&gt;Joined record fields (Employee)&lt;br&gt;■ customer.salesrep.&lt;joinedFieldName&gt;</td>
</tr>
</tbody>
</table>
Join Trail Formatting

When you use join trails to access fields in joined queries, you can add whitespace characters and parentheses to improve the readability of your formulas. For example, consider this join trail:

<table>
<thead>
<tr>
<th>Join Type</th>
<th>Join Operation</th>
<th>Join Trail</th>
</tr>
</thead>
</table>
| Polymorphic using `Query.joinTo(options)` or `Component.joinTo(options)` | // The base record type is Transaction 
 var myTransactionQuery = query.create({
   type: query.Type.TRANSACTION 
 }); 
 // The joined record type is Employee 
 var myEmployeeJoin = myTransactionQuery.joinTo({
   fieldId: 'createdby', 
   target: 'employee' 
 }); | Base record fields (Transaction) 
 - transaction.<baseFieldName> 
   Example:## transaction.entity 
 Joined record fields (Employee) 
 - transaction.#<baseFieldName>^employee.#<joinedFieldName> 
   Example:## transaction.#<createdby>^employee.#email |
| Inverse using `Query.joinFrom(options)` or `Component.joinFrom(options)` | // The base record type is Employee 
 var myEmployeeQuery = query.create({
   type: query.Type.EMPLOYEE 
 }); 
 // The joined record type is Transaction 
 var myTransactionJoin = myEmployeeQuery.joinFrom({
   fieldId: 'entity', 
   source: 'transaction' 
 }); | Base record fields (Employee) 
 - employee.<baseFieldName> 
   Example:## employee.entityid 
 Joined record fields (Transaction) 
 - employee.#<baseFieldName>^transaction.#<daysoverdue> 
   Example:## employee.#entity^transaction.#<daysoverdue> |

Example: `customer.salesrep.phone`
employee.entity < transaction.daysoverdue

The following join trails are equivalent to this one:

- employee.entity < transaction.daysoverdue
- (employee.entity) < (transaction.daysoverdue)

Relative Dates in the N/query Module

You can use relative dates when you create query conditions. When you use a query.RelativeDate object or query.RelativeDateRange enum value to create a query condition, make sure that you use an operator that makes sense for the date range that you provide to Query.createCondition(options) or Component.createCondition(options). The query.Operator enum contains the supported operators for the N/query module, but not all operators apply to relative dates. Use the following operators with relative dates:

- AFTER
- AFTER_NOT
- BEFORE
- BEFORE_NOT
- ON
- ON_NOT
- ON_OR_AFTER
- ON_OR_AFTER_NOT
- ON_OR_BEFORE
- ON_OR_BEFORE_NOT
- WITHIN
- WITHIN_NOT

When you create a query condition using the WITHIN or WITHIN_NOT operators and a query.RelativeDate object, the condition uses the current date as one of the boundaries of the date range. For example, consider the following query.RelativeDate object that represents dates from two days ago until the current date:

```javascript
var myDatesAgo = query.createDateRange({
    dateRangeId: query.DateRangeId.DAYS_AGO,
    value: 2
});
```

You can use this `myDatesAgo` object when you create a query condition. Consider the following query condition that is created using the WITHIN operator and this `myDatesAgo` object:

```javascript
var myCondition = myQuery.createCondition({
    fieldId: ‘trandate’,
    operator: query.Operator.WITHIN,
    values: myDatesAgo
});
```

This query condition matches dates that are between two days ago and the current date (the day before yesterday, yesterday, and today).
Conversely, consider the following `query.RelativeDate` object that represents dates from the current date until two days from now:

```javascript
var myDatesFromNow = query.createDateRange({
    dateRangeId: query.DateRangeId.DAYS_FROM_NOW,
    value: 2
});
```

If you create a query condition using the `WITHIN` operator and this `myDatesFromNow` object, the condition matches dates that are between the current date and two days from now (today, tomorrow, and the day after tomorrow).

You can use the `query.RelativeDate` object, the `query.RelativeDateRange` enum, and the `WITHIN` operator to specify complex date ranges. You can do this in several ways:

- **Use a single `query.RelativeDate` object or `query.RelativeDateRange` enum value.** When you use a single `query.RelativeDate` object, the current date is used automatically as one of the boundaries, as discussed earlier. When you use a single `query.RelativeDateRange` enum value, the `start` and `end` properties of the date range are used automatically as the boundaries. For example:

  ```javascript
  var myComplexCondition = myQuery.createCondition({
    fieldId: 'trandate',
    operator: query.Operator.WITHIN,
    values: query.RelativeDateRange.SAME_DAY_LAST_WEEK
  });
  ```

  In this example, the first boundary is the beginning of the same day last week, and the second boundary is the end of the same day last week. Using `query.RelativeDateRange.SAME_DAY_LAST_WEEK` is equivalent to using either of the following:

  - `query.RelativeDateRange.SAME_DAY_LAST_WEEK.interval`
  - `[query.RelativeDateRange.SAME_DAY_LAST_WEEK.start, query.RelativeDateRange.SAME_DAY_LAST_WEEK.end]`

- **Use the `start` and `end` properties of values in the `query.RelativeDateRange` enum directly in the `values` parameter for `Query.createCondition(options)` or `Component.createCondition(options)`.** For example:

  ```javascript
  var myComplexCondition = myQuery.createCondition({
    fieldId: 'trandate',
    operator: query.Operator.WITHIN,
    values: [query.RelativeDateRange.THIS_FISCAL_YEAR.start, query.RelativeDateRange.YESTERDAY.end]
  });
  ```

- **Use a combination of `query.RelativeDateRange` enum values and custom `query.RelativeDate` objects.** For example:

  ```javascript
  var myEndDate = query.createDateRange({
    dateRangeId: query.DateRangeId.WEEKS_AGO,
    value: 2
  });

  var myComplexCondition = myQuery.createCondition({
    fieldId: 'trandate',
    operator: query.Operator.WITHIN,
    values: [query.RelativeDateRange.THREE_FISCAL_YEARS_AGO.start, myEndDate]
  });
  ```
query.Column

Object Description

Encapsulates a query result column.

The `query.Column` object is the equivalent of the `search.Column` object in the N/search Module. The `query.Column` object describes the field types (columns) that are displayed from the query results.

To create columns:

- Use `Query.createColumn(options)` to create a column on the initial query definition created with `query.create(options)`.
- Use `Component.createColumn(options)` to create a column on a join relationship created with `Query.autoJoin(options)` or `Component.autoJoin(options)`.
- Assign all created columns as array values to `Query.columns`. For an example, see Syntax.

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/query Module

Methods and Properties

Column Object Members

Since 2018.1

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    })
];
```
Column.aggregate

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes an aggregate function that is performed on the query result column. An aggregate function performs a calculation on the column values and returns a single value.</td>
<td></td>
</tr>
<tr>
<td>This property is set when <code>Query.createColumn(options)</code> or <code>Component.createColumn(options)</code> is executed.</td>
<td></td>
</tr>
<tr>
<td>For a list of supported aggregate functions, see the <code>query.Aggregate</code> enum.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/query Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Parent Object</th>
<th>query.Column</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Sibling Object</th>
<th>Column Object Members</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2018.1</th>
</tr>
</thead>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/query Module Script Samples`.

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myAggColumn = myTransactionQuery.createColumn({
    fieldId: 'amount',
    aggregate: query.Aggregate.AVERAGE
});
```
myTransactionQuery.columns = [myAggColumn];

var theAggregate = myAggColumn.aggregate;
...
// Add additional code

Column.component

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Holds a reference to the query.Component object to which this query result column belongs. This property is set when Query.createColumn(options) or Component.createColumn(options) is executed.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>query.Component object (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Column</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Column Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myAmountColumn = myTransactionQuery.createColumn({
    fieldId: 'amount'
});

var theComponent = myAmountColumn.component;
...
// Add additional code
```

Column.consolidated

Indicates whether the query result column displays consolidated currency amounts. This property is set when Query.createColumn(options) or Component.createColumn(options) is executed. The default value is false.

Consolidation is a financial concept of converting currency amounts from the base currency of a child subsidiary to the base currency of its parent subsidiary. If the value of this property is true, the column displays currency amounts in the base currency for the parent subsidiary. If the value of this property is false, the column displays currency amounts in the base...
currency of the child subsidiary. For more information about consolidation, see the help topic Consolidated Exchange Rates.

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Column</td>
</tr>
<tr>
<td>Sibling Object</td>
<td>Column Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

## Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myTranLinesJoin = myTransactionQuery.autoJoin({
    fieldId: 'transactionlines'
});

myTransactionQuery.condition = myTranLinesJoin.createCondition({
    fieldId: 'netamount',
    operator: query.Operator.GREATER,
    values: 50000
});

myTransactionQuery.columns = [
    myTranLinesJoin.createColumn({
        fieldId: 'netamount'
    })
];

var unconsolidatedResultSet = myTransactionQuery.run();

// Log unconsolidated amounts
for (var i in unconsolidatedResultSet.results) {
    log.debug(unconsolidatedResultSet.results[i].values[0]);
}

myTransactionQuery.columns = [
    myTranLinesJoin.createColumn({
        fieldId: 'netamount',
        consolidated: true
    })
];

var consolidatedResultSet = myTransactionQuery.run();
```
// Log consolidated amounts
for (var i in consolidatedResultSet.results)
    log.debug(consolidatedResultSet.results[i].values[0]);
...
// Add additional code

Column.fieldId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds the name of the query result column.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is set during the execution of Query.createColumn(options) or Component.createColumn(options). This property and the Column.formula property cannot be set at the same time.</td>
</tr>
</tbody>
</table>

| Type                | string (read-only) |

| Module              | N/query Module     |

| Parent Object       | query.Column       |

| Sibling Object      | Column Object Members |

| Since               | 2018.1 |

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myAmountColumn = myTransactionQuery.createColumn({
    fieldId: ‘amount’
});

var theFieldId = myAmountColumn.fieldId;
...
// Add additional code

Column.formula

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Describes a formula used to create the query result column.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is set during the execution of Query.createColumn(options) or Component.createColumn(options). This property and the Column.fieldId property cannot be set at the same time.</td>
</tr>
<tr>
<td></td>
<td>For more information on formulas, see the help topics Formulas in Search and SQL Expressions.</td>
</tr>
</tbody>
</table>
### Type

String (read-only)

#### Module

N/query Module

#### Parent Object

query.Column

#### Sibling Object Members

Column Object Members

#### Since

2018.1

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myFormulaColumn = myTransactionQuery.createColumn({
    type: query.ReturnType.CURRENCY,
    formula: '{amount} * 125'
});

var theFormula = myFormulaColumn.formula;
...
// Add additional code
```

### Column.groupBy

**Property Description**

Indicates whether the query results are grouped by this query result column. This property is set during the execution of `Query.createColumn(options)` or `Component.createColumn(options)`.

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Column</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Column Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
...
```
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var myGroupByColumn = myCustomerQuery.createColumn({
    fieldId: 'currency',
    groupBy: true
});

var theGroupBy = myGroupByColumn.groupBy;
...

// Add additional code
```

### Column.type

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes the return type of the formula used to create the query result column.</td>
<td></td>
</tr>
<tr>
<td>This property is set during the execution of <code>Query.createColumn(options)</code> or <code>Component.createColumn(options)</code>. If a formula is specified when these methods are called, this property contains the return type of the formula. If a formula is not specified, this property is null.</td>
<td></td>
</tr>
<tr>
<td>For more information on formulas, see the help topics Formulas in Search and SQL Expressions.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query:Column</td>
</tr>
<tr>
<td>Sibling Object</td>
<td>Column Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myFormulaColumn = myTransactionQuery.createColumn({
    type: query.ReturnType.CURRENCY,
    formula: '{amount} * 125'
});

var theFormulaType = myFormulaColumn.type;
...
// Add additional code
```
query.Component

**Object Description**

Encapsulates one component of the query definition. Each new component is created as a child to the previous component. All components exist as children to the query definition (query.Query).

You can think of a component as a building block; each new component builds on the previous component created. The last component created encapsulates the relationship between it and all of its parent components.

The query definition always contains at least one component. Queries with joins contain multiple components. The query definition (query.Query) contains a child query.Component object for each of the following:

- **The initial query definition**: The initial query.Component object is called the root component. It encapsulates the initial search type passed to query.create(options). The root component is automatically created with the query.Query object and is a child of the query.Query object. The Query.root property contains a reference to the root component.

- **The first join**: The second query.Component object is created with Query.autoJoin(options). It encapsulates the relationship between the initial query definition and the second search type. This relationship is determined by the join ID passed to Query.autoJoin(options). The second query.Component object is a child of the root component.

- **Each subsequent join**: The third query.Component object is created with Component.autoJoin(options). All subsequent joins and their respective query.Component objects are also created with Component.autoJoin(options). Each of these query.Component objects encapsulates the relationship between all previous search types and the new search type. This relationship is determined by the join ID passed to Component.autoJoin(options).

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/query Module

**Methods and Properties**

Component Object Members

**Since**

2018.1

**Syntax**

⚠️ **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
  type: query.Type.CUSTOMER
});

var salesrep = search.join({
  fieldId: 'salesrep'
});

search.columns = [
  search.createColumn({
    fieldId: 'entityid'
  }],
```
Component.autoJoin(options)

Method Description

Creates a join relationship.

Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type.

After you create the initial query definition, use Query.autoJoin(options) to create your first join (query.Component). Then use Component.autoJoin(options) to create each subsequent join (query.Component).

⚠️ Important: The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic Available Records.

Returns

query.Component object

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/query Module

Parent Object

query.Component

Sibling Object Members

Component Object Members

Since

2018.2
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The column type (field type) that joins the parent component to the new component. Obtain this value from the Records Browser:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Go to the parent component's record type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Scroll until you see the Search Joins table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Locate the appropriate value in the Join ID column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATIONSHIP_ALREADY_USED</td>
<td>The specified join relationship already exists.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
  type: query.Type.TRANSACTION
});

var entity = search.autoJoin({
  fieldId: 'entity'
});

search.columns = [entity.createColumn({
  fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
  column: search.columns[0],
  ascending: false
})];

var results = search.runPaged({
  pageSize: 10
});
```

Component.createColumn(options)

Method Description: Creates a query result column based on the query.Component object.
The `query.Column` object is the equivalent of the `search.Column` object in the `N/search Module`. The `query.Column` object describes the field types (columns) that are displayed from the query results.

To create columns:

- Use `Component.createColumn(options)` to create conditions on the join relationships created with `Query/autoJoin(options)` and `Component/autoJoin(options)`. Use this method in one of two ways:
  - Pass in an argument for the parameter `options.fieldId`.
  - Pass in an argument for the parameter `options.formula`. If you use this option, you can also use the optional parameter `options.type`.
- If needed, use `Query/createColumn(options)` to create columns on the initial query definition created with `query.create(options)`.
- Assign all created columns as array values to `Query/columns`. For an example, see Syntax.

Returns

`query.Column` object

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Governance**

None

**Module**

`N/query Module`

**Parent Object**

`query.Component`

**Sibling Object Members**

Component Object Members

**Since**

2018.1

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.fieldId | string   | required if options.formula is not used | The name of the query result column. This value sets the `Column.fieldId` property. Obtain this value from the Records Browser:  
1. Go to the appropriate record type.  
2. Scroll until you see the Search Columns table.  
3. Locate the appropriate value in the Internal ID column.  
For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser. |
<p>| options.formula | string   | required if options.fieldId is not used | The formula used to create the query result column. This value sets the <code>Column.formula</code> property. For more information on formulas, see the help topics Formulas in Search and SQL Expressions. |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>optional if options.formula is used</td>
<td>If you use the options.formula parameter, use this parameter to explicitly define the formula's return type. Defining the formula's return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the Column.type property. Use the appropriate query.ReturnType enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
<tr>
<td>options.aggregate</td>
<td>string</td>
<td>optional</td>
<td>Use this parameter to run an aggregate function on your query result column. An aggregate function performs a calculation on the column values and returns a single value. This value sets the Column.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
<tr>
<td>options.consolidated</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether the query result column shows consolidated currency amounts. This value sets the Column.consolidated property. If you do not pass in an argument, the default value is set to false.</td>
</tr>
<tr>
<td>options.groupBy</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether the query results are grouped by this query result column. This value sets the Column.groupBy property. If you do not pass in an argument, the default value is set to false.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
  type: query.Type.CUSTOMER
});

var salesrep = search.join({
  fieldId: 'salesrep'
});

search.columns = [
  search.createColumn({
    fieldId: 'entityid'
  }),
  search.createColumn({
    fieldId: 'id'
  }),
  salesrep.createColumn({
    fieldId: 'entityid'
  }),
];
```
Component.createCondition(options)

**Method Description**

Creates a condition (query filter) based on the query.Component object.

A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects.

To create conditions:

- Use `Component.createCondition(options)` to create conditions on the join relationships created with `Query.autoJoin(options)` and `Component.autoJoin(options)`. Use this method in one of two ways:
  - Pass in arguments for the parameters `options.fieldId`, `options.operator`, and `options.values`. The combination of these arguments translates to `<filter column><operator><field value>` (for example, 'city' equals 'Boston').
  - Pass in an argument for the parameter `options.formula`. If you use this option, you can also use the optional parameter `options.type`.
- If needed, use `Query.createCondition(options)` to create conditions on the initial query definition created with `query.create(options)`.
- If you have multiple conditions, use them to create a new nested condition with the methods `Query.and()`, `Query.or()`, and `Query.not()`.
- Assign your simple or nested condition to `Query.condition`. For an example, see Syntax.

**Returns**

query.Condition object

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/query Module

**Parent Object**

query.Component

**Sibling Object Members**

Component Object Members
### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required if options.operator and options.values are used</td>
<td>The name of the condition. This value sets the Condition.fieldId property. Obtain this value from the Records Browser: 1. Go to the appropriate record type. 2. Scroll until you see the Search Filters table. 3. Locate the appropriate value in the Internal ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
<tr>
<td>options.operator</td>
<td>string</td>
<td>required if options.fieldId and options.values are used</td>
<td>The operator used by the condition. This value sets the Condition.operator property. Use the appropriate query.Operator enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
<tr>
<td>options.values</td>
<td>string[]</td>
<td>required if options.fieldId and options.operator are used</td>
<td>An array of string values. This value sets the Condition.values property.</td>
</tr>
<tr>
<td>options.formula</td>
<td>string</td>
<td>required if options.fieldId, options.operator, and options.values are not used</td>
<td>The formula used to create the condition. This value sets the Condition.formula property. For more information on formulas, see the help topics Formulas in Search and SQL Expressions.</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>optional if options.formula is used</td>
<td>If you use the options.formula parameter, use this parameter to explicitly define the formula's return type. Defining the formula's return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the Condition.type property. Use the appropriate query.ReturnType enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
<tr>
<td>options.aggregate</td>
<td>string</td>
<td>optional</td>
<td>Use this parameter to run an aggregate function on a condition. An aggregate function performs a calculation on the condition values and returns a single value. This value sets the Condition.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create(
    type: query.Type.CUSTOMER
);

var salesrep = search.join(
    { fieldId: 'salesrep'
    });
var location = salesrep.join(
    { fieldId: 'location'
    });

var cond1 = search.createCondition(
    { fieldId: 'id',
      operator: query.Operator.EQUAL,
      values: 107
    });
var cond2 = search.createCondition(
    { fieldId: 'id',
      operator: query.Operator.EQUAL,
      values: 2647
    });
var cond3 = salesrep.createCondition(
    { fieldId: 'email',
      operator: query.Operator.START_WITH_NOT,
      values: 'foo'
    });

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```

Component.createSort(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a sort based on the query.Component object. The query.Sort object describes a sort that is placed on a particular query result column or condition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>To create a sort:</td>
<td>- Use <code>Component.createSort(options)</code> to create a sort based on a join relationship created with <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code>.</td>
</tr>
<tr>
<td></td>
<td>- Use <code>Query.createSort(options)</code> to create a sort based on the initial query definition created with <code>query.create(options)</code>.</td>
</tr>
<tr>
<td></td>
<td>- Assign all created sorts as array values to <code>Query.sort</code>. For an example, see Syntax.</td>
</tr>
</tbody>
</table>

| Returns            | query.Sort                                                                 |
```
Supported Script Types  
Client and server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance  
None

Module  
N/query Module

Parent Object  
query.Component

Sibling Object Members  
Component Object Members

Since  
2018.1

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.column</td>
<td>query.Column</td>
<td>required</td>
<td>The query result column that you want to sort by. This value sets the Sort.column property.</td>
</tr>
<tr>
<td>options.ascending</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether the sort direction is ascending. This value sets the Sort.ascending property. The default value of this property is true, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to false.</td>
</tr>
<tr>
<td>options.caseSensitive</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether the sort is case sensitive. This value sets the Sort.caseSensitive property.</td>
</tr>
</tbody>
</table>

If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same. For example, the following list of items is sorted using a case-sensitive sort with a sort direction of ascending:

- Banana  
- Orange  
- apple  
- grapefruit  
- kiwi

Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending:

- apple  
- Banana  
- grapefruit  
- kiwi  
- Orange
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.locale</td>
<td>string</td>
<td>optional</td>
<td>The locale to use for the sort. This value sets the Sort.locale property. A locale represents a combination of language and region, and it can affect how certain values (such as strings) are sorted. Use this property to ensure that query results are sorted using locale-specific rules. Use the appropriate query.SortLocale enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
<tr>
<td>options.nullsLast</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether query results with null values are listed at the end of the query results. This value sets the Sort.nullsLast property. The default value of this property is the value of the options.ascending property. For example, if the options.ascending property is set to true, the options.nullsLast property is also set to true.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create(
   type: query.Type.CUSTOMER
);

var salesrep = search.join(
   fieldId: 'salesrep'
);

search.columns = [
   search.createColumn(
      fieldId: 'entityid'
   ),
   search.createColumn(
      fieldId: 'id'
   ),
   salesrep.createColumn(
      fieldId: 'entityid'
   ),
   salesrep.createColumn(
      fieldId: 'email'
   ),
   salesrep.createColumn(
      fieldId: 'hiredate'
   ),
];
```
search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];
var resultSet = search.run();

### Component.join(options)

**Method Description**

Creates a join relationship. This method is an alias to `Component.autoJoin(options)`.

Use the method `query.create(options)` to create your initial query definition (`query.Query`). The initial query definition uses one search type. For available search types, see `query.Type`.

After you create the initial query definition, use `Query.autoJoin(options)` to create your first join (`query.Component`). Then use `Component.join(options)` to create each subsequent join (`query.Component`).

⚠️ **Important:** The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic `Available Records`.

**Returns**

`query.Component` object

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Governance**

None

**Module**

N/query Module

**Parent Object**

`query.Component`

**Sibling Object Members**

Component Object Members

**Since**

2018.1

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The column type (field type) that joins the parent component to the new component. This value determines the columns on which the components are joined and the type of the newly joined component. Obtain this value from the <code>Records Browser:</code></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
|           |      |                      | 1. Go to the parent component's record type.  
2. Scroll until you see the Search Joins table.  
3. Locate the appropriate value in the Join ID column.  
For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser. |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.TRANSACTION
});
var entity = search.join({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

**Component.joinFrom(options)**

**Method Description**

Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the `Component.source` property on the returned `query.Component` object.

Use the method `query.create(options)` to create your initial query definition (`query.Query`). The initial query definition uses one search type. For available search types, see `query.Type`.

After you create the initial query definition, use this method to create explicit directional joins from other components to this component.

**Important:** The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic Available Records.

**Returns**

`query.Component` object
Supported Script Types  Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance  None

Module  N/query Module

Parent Object  query.Component

Sibling Object Members  Component Object Members

Since  2018.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The column type (field type) that joins the parent component to the new component. Obtain this value from the Records Browser: 1. Go to the parent component's record type. 2. Scroll until you see the Search Joins table. 3. Locate the appropriate value in the Join ID column. For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
<tr>
<td>options.source</td>
<td>string</td>
<td>required</td>
<td>The search type of the component joined to this component. This value sets the Component.source property. This value can be described as the inverse relationship of this component, and it determines the source search type of the newly joined component.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATIONSHIP_ALREADY_USED</td>
<td>The specified join relationship already exists.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
var search = query.create({
   type: query.Type.EMPLOYEE
});

var salesorder = search.joinFrom({
   fieldId: ‘salesrep’,
});
```
source: 'salesorder'
});

var items = salesorder.autoJoin({
    fieldId: 'item'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'hiredate'
    }),
    salesorder.createColumn({
        fieldId: 'id'
    }),
    salesorder.createColumn({
        fieldId: 'trandate'
    })
];

var sort1 = search.createSort({
    column: search.columns[0],
    ascending: false
});
var sort2 = search.createSort({
    column: search.columns[1],
    ascending: true
});
search.sort = [sort1, sort2];

var results = search.run();

---

## Component.joinTo(options)

### Method Description

Creates an explicit directional join relationship to another component from this component (a polymorphic join). This method sets the `Component.target` property on the returned `query.Component` object.

Use the method `query.create(options)` to create your initial query definition (`query.Query`). The initial query definition uses one search type. For available search types, see `query.Type`.

After you create the initial query definition, use this method to create explicit directional joins to other components from this component.

**Important:** The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic `Available Records`.

### Returns

`query.Component` object

### Supported Script Types

Client and server-side scripts

For more information, see the help topic `SuiteScript 2.0 Script Types`. 

---
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.fieldId | string  | required            | The column type (field type) that joins the parent component to the new component.  
|               |         |                     | Obtain this value from the Records Browser:                                   |
|               |         |                     | 1. Go to the parent component's record type.                                    |
|               |         |                     | 2. Scroll until you see the Search Joins table.                                 |
|               |         |                     | 3. Locate the appropriate value in the Join ID column.                           |
|               |         |                     | For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser. |
| options.target | string  | required            | The search type of the component joined to this component. This value sets the Component.target property.  
|               |         |                     | This value can be described as the polymorphic relationship of this component, and it determines the target search type of the newly joined component. |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATIONSHIP_ALREADY_USED</td>
<td>The specified join relationship already exists.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
  type: query.Type.TRANSACTION
});

var entity = search.joinTo({
  fieldId: 'entity',
  target: query.Type.CUSTOMER
});
```
search.columns = [
    entity.createColumn({
        fieldId: 'subsidiary'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false
    })
];

var results = search.runPaged({
    pageSize: 10
});

---

**Component.child**

**Property Description**
Holds a references to children of this component. The value of this property is an object of key/value pairs. Each key is the name of a child component. Each respective value refers to the corresponding `query.Component` object.

The object values are set during the execution of `Query.autoJoin(options)` and `Component.autoJoin(options)`. The order of the key/value pairs reflects the parent/child hierarchy.

**Type**
Object (read-only)

**Module**
N/query Module

**Parent Object**
query.Component

**Sibling Object**
Component Object Members

**Since**
2018.1

---

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myDeptJoin = mySalesRepJoin.autoJoin({
    fieldId: 'department'
});
```
```javascript
var theChild = mySalesRepJoin.child;
...
// Add additional code
```

**Component.parent**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds a references to the parent <code>query.Component</code> object of this component. This property is set during the execution of <code>Query.autoJoin(options)</code> or <code>Component.autoJoin(options)</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td><code>N/query Module</code></td>
</tr>
<tr>
<td>Parent Object</td>
<td><code>query.Component</code></td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td><code>Component Object Members</code></td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/query Module Script Samples`.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myDeptJoin = mySalesRepJoin.autoJoin({
    fieldId: 'department'
});

var theParent = myDeptJoin.parent;
...
// Add additional code
```

**Component.source**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Describes the search type of the component joined to this component. This property can also be described as the inverse relationship of this component. This property is set during the execution of <code>Query.joinFrom(options)</code> and <code>Component.joinFrom(options)</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
var myEmployeeQuery = query.create({
  type: query.Type.EMPLOYEE
});

var myTransactionJoin = myEmployeeQuery.joinFrom({
  fieldId: 'entity',
  source: 'transaction'
});

var theSource = myTransactionJoin.source;
...
// Add additional code
```

### Component.target

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes the search type of this component. This property can also be described as the polymorphic relationship of this component. This property is set during the execution of <code>Query.joinTo(options)</code> and <code>Component.joinTo(options)</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Component</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Component Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>
Component.type

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes the search type of this component.</td>
<td></td>
</tr>
<tr>
<td>This property is set during the execution of <code>query.autoJoin(options)</code> and <code>Component.autoJoin(options)</code>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td><code>N/query Module</code></td>
</tr>
<tr>
<td>Parent Object</td>
<td><code>query.Component</code></td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td><code>Component Object Members</code></td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/query Module Script Samples`.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});
var theType = myCustomerQuery.type;
... // Add additional code
```

query.Condition

**Object Description**

A condition narrows the query results. The `query.Condition` object acts in the same capacity as the `search.Filter` object in the `N/search Module`. The primary difference is that `query.Condition` objects can contain other `query.Condition` objects.

To create conditions:

- Use `Query.createCondition(options)` to create conditions for the initial query definition created with `query.create(options)`.
Use `Component.createCondition(options)` to create conditions for the join relationships created with `Query.autoJoin(options)` and `Component.autoJoin(options)`.

If you have multiple conditions, use them to create a new nested condition with the methods `Query.and()`, `Query.or()`, and `Query.not()`.

Assign your simple or nested condition to `Query.condition`. For an example, see Syntax.

### Syntax

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});
var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});
var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});
search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);
var resultSet = search.run();
```
### Condition.aggregate

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes an aggregate function that is performed on the condition. An aggregate function performs a calculation on the condition values and returns a single value. This property is set during the execution of <code>Query.createCondition(options)</code> or <code>Component.createCondition(options)</code>.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** This property is not applicable to parent conditions created with the execution of `Query.and()`, `Query.or()`, or `Query.not()`.

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Condition</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Condition Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var myAggregateCondition = myCustomerQuery.createCondition({
    fieldId: 'openingbalance',
    operator: query.Operator.GREATER,
    values: 10000,
    aggregate: query.Aggregate.MAXIMUM
});

var theAggregate = myAggregateCondition.aggregate;
...
// Add additional code
```

### Condition.children

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holds an array of child conditions used to create the parent condition.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** This property is applicable to only parent conditions created with the execution of `Query.and()`, `Query.or()`, or `Query.not()`.

<table>
<thead>
<tr>
<th>Type</th>
<th>query.Condition[]</th>
</tr>
</thead>
</table>
## Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var myFirstCondition = myCustomerQuery.createCondition({
    fieldId: 'openingbalance',
    operator: query.Operator.GREATER,
    values: 10000
});

var mySecondCondition = myCustomerQuery.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

var myComplexCondition = myCustomerQuery.and(myFirstCondition, mySecondCondition);

var theChildren = myComplexCondition.children;
...
// Add additional code
```

### Condition.component

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes the component used to created the condition</td>
<td>This property is set during the execution of <code>Query.createCondition(options)</code> and <code>Component.createCondition(options)</code>.</td>
</tr>
<tr>
<td>Note: This property is not applicable to parent conditions created with the execution of <code>Query.and()</code>, <code>Query.or()</code>, or <code>Query.not()</code></td>
<td></td>
</tr>
</tbody>
</table>

| Type | string (read-only) |
| Module | N/query Module |
| Parent Object | query.Condition |
| Sibling Object Members | Condition Object Members |
Since 2018.1

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
  fieldId: 'salesrep'
});

var myCondition = mySalesRepJoin.createCondition({
  fieldId: 'email',
  operator: query.Operator.START_WITH,
  values: 'mentor'
});

var theComponent = myCondition.component;
...
// Add additional code
```

Condition.fieldId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds the name of the condition.</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property is set during the execution of Query.createCondition(options) and Component.createCondition(options).</td>
<td></td>
</tr>
</tbody>
</table>

⚠️ Note: This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Condition</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Condition Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
```
var myCustomerQuery = query.create(
    type: query.Type.CUSTOMER
));

var myCondition = myCustomerQuery.createCondition({
    fieldId: 'openingbalance',
    operator: query.Operator.GREATER,
    values: 10000
});

var theFieldId = myCondition.fieldId;
...

// Add additional code

var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myFormulaCondition = myTransactionQuery.createCondition({
    formula: '{amount} * 125',
    operator: query.Operator.GREATER,
    values: 50000,
    type: query.ReturnType.CURRENCY
});

// Add additional code
```javascript
var theFormula = myFormulaCondition.formula;
...
// Add additional code
```

## Condition.operator

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds the name of the operator used to create the condition.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is set during the execution of <code>Query.createCondition(options)</code> and <code>Component.createCondition(options)</code>.</td>
</tr>
</tbody>
</table>

**Note:** This property is not applicable to parent conditions created with the execution of `Query.and()`, `Query.or()`, or `Query.not()`.

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Condition</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Condition Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

var myCondition = myCustomerQuery.createCondition({
  fieldId: 'openingbalance',
  operator: query.Operator.GREATER,
  values: 10000
});

var theOperator = myCondition.operator;
...
// Add additional code
```

## Condition.type

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The return type of the formula used to create the condition.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is set during the execution of <code>Query.createCondition(options)</code> or <code>Component.createCondition(options)</code>.</td>
</tr>
</tbody>
</table>
For more information on formulas, see the help topics Formulas in Search and SQL Expressions.

Note: This property is not applicable to parent conditions created with the execution of Query.and(), Query.or(), or Query.not().

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Condition</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Condition Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myFormulaCondition = myTransactionQuery.createCondition({
    formula: '{amount} * 125',
    operator: query.Operator.GREATER,
    values: 50000,
    type: query.ReturnType.CURRENCY
});

var theFormulaType = myFormulaCondition.type;
...
// Add additional code
```

### Condition.values

**Property Description**

Holds an array of values used by an operator to create the condition.

This property is set by passing in values for `options.fieldId`, `options.operator` and `options.values` during the execution of `Query.createCondition(options)` or `Component.createCondition(options)`.

Note: This property is not applicable to parent conditions created with the execution of `Query.and()`, `Query.or()`, or `Query.not()`.

<table>
<thead>
<tr>
<th>Type</th>
<th>string[] (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Condition</td>
</tr>
</tbody>
</table>
query.Page

Object Description
One page of the paged query results.

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/query Module

Methods and Properties
Page Object Members

Since
2018.1

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var myCondition = myCustomerQuery.createCondition({
    fieldId: 'firstname',
    operator: query.Operator.ANY_OF,
    values: ['Martin', 'Russell', 'Janina']
});

var theValues = myCondition.values;
...
// Add additional code
```
myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'firstname'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

// Fetch results using an iterator
var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    log.debug(currentPage.pageRange.size);
    return true;
});

// Alternatively, fetch results using a loop
for (var i = 0; i < myPagedResults.pageRanges.length; i++) {
    var currentPage = myPagedResults.fetch(i);
    log.debug(currentPage.pageRange.size);
}
...

// Add additional code

---

**Page.data**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>References the query results contained in this page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>query.ResultSet (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Page</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Page Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.
var myCustomerQuery = query.create({
        type: query.Type.CUSTOMER
    });

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }]
];

var myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var theData = currentPage.data;
    return true;
});

// Add additional code

Page.isFirst

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether the page is the first of the paged query results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Page</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Page Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

// Add additional code
...

var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    })
];
```javascript
var myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var isFirst = currentPage.isFirst;
    return true;
});
...
// Add additional code

Page.isLast

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether the page is the last of the paged query results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Page</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Page Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```
Page.pageRange

**Property Description**
The range of query results for this page.

**Type**
query.PageRange (read-only)

**Module**
N/query Module

**Parent Object**
query.Page

**Sibling Object Members**
Page Object Members

**Since**
2018.1

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
  myCustomerQuery.createColumn({
    fieldId: 'entityid'
  })
];

var myPagedResults = myCustomerQuery.runPaged({
  pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
  var currentPage = resultPage.value;
  var isLast = currentPage.isLast;
  return true;
});
... // Add additional code
```
Page.pagedData

**Property Description**  References the set of paged query results that this page is from.

**Type**  
query.PagedData (read-only)

**Module**  
N/query Module

**Parent Object**  
query.Page

**Sibling Object Members**  
Page Object Members

**Since**  
2018.1

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    })
];

var myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var thePagedData = currentPage.pagedData;
    return true;
});
...
// Add additional code
```

**query.PagedData**

**Object Description**  
Encapsulates a set of paged query results. This object also contains information about the set of paged results it encapsulates.

Use `Query.runPaged()` or `Query.runPaged.promise()` to create this object.
## Supported Script Types

Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](https://www.netsuite.com/).  

## Module

N/query Module

## Methods and Properties

PagedData Object Members

## Since

2018.1

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](https://www.netsuite.com/).

```javascript
// Add additional code
...
var myCustomerQuery = query.create(
    type: query.Type.CUSTOMER
);

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'firstname'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var myPagedResults = myCustomerQuery.runPaged(
    pageSize: 10
);

// Fetch results using an iterator
var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var currentPagedData = currentPage.pagedData;
    log.debug(currentPage.pageRange.size);
    return true;
});

// Alternatively, fetch results using a loop
for (var i = 0; i < myPagedResults.pageRanges.length; i++) {
    var currentPage = myPagedResults.fetch(i);
    var currentPagedData = currentPage.pagedData;
    log.debug(currentPage.pageRange.size);
}
...
// Add additional code
PagedData.iterator()

Method Description | Standard SuiteScript 2.0 object for iterating through results
Returns | Iterator object
Supported Script Types | Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.
Governance | None
Module | N/query Module
Parent Object | query.PagedData
Sibling Object Members | PagedData Object Members
Since | 2018.1

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
  myCustomerQuery.createColumn({
    fieldId: 'entityid'
  })
];

var myPagedResults = myCustomerQuery.runPaged({
  pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
  var currentPage = resultPage.value;
  var currentPagedData = currentPage.pagedData;
  return true;
});
...
// Add additional code
```

PagedData.count

Property Description | Describes the total number of paged query result rows.
## Type

**number (read-only)**

**Module**

N/query Module

**Parent Object**

query.PagedData

**Sibling Object Members**

PagedData Object Members

**Since**

2018.1

---

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
var myCustomerQuery = query.create(
  type: query.Type.CUSTOMER
);)

myCustomerQuery.columns = [
  myCustomerQuery.createColumn(
    fieldId: 'entityid'
  )
];

var myPagedResults = myCustomerQuery.runPaged(
  pageSize: 10
);

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
  var currentPage = resultPage.value;
  var currentPagedData = currentPage.pagedData;
  var theCount = currentPagedData.count;
  return true;
});
...
// Add additional code
```

---

### PagedData.pageRanges

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds an array of page ranges for the paged query results.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>query.PageRange[]</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/query Module</td>
</tr>
<tr>
<td><strong>Parent Object</strong></td>
<td>query.PagedData</td>
</tr>
<tr>
<td><strong>Sibling Object Members</strong></td>
<td>PagedData Object Members</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2018.1</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    })
];

var myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var currentPagedData = currentPage.pagedData;
    var thePageRanges = currentPagedData.pageRanges;
    return true;
});
...
// Add additional code
```

### PagedData.pageSize

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PagedData.pageSize</td>
<td>Describes the number of query result rows per page.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>number (read-only)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/query Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Parent Object</th>
<th>query.PagedData</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Sibling Object Members</th>
<th>PagedData Object Members</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2018.1</th>
</tr>
</thead>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
```
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
  myCustomerQuery.createColumn({
    fieldId: 'entityid'
  })
];

var myPagedResults = myCustomerQuery.runPaged({
  pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
  var currentPage = resultPage.value;
  var currentPagedData = currentPage.pagedData;
  var thePageSize = currentPagedData.pageSize;
  return true;
});

// Add additional code

query.PageRange

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the range of query results for a page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Methods and Properties</td>
<td>PageRange Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
  myCustomerQuery.createColumn({
    fieldId: 'entityid'
  })
];
```
myCustomerQuery.createColumn({
    fieldId: 'firstname'
});
myCustomerQuery.createColumn({
    fieldId: 'email'
});

var myPagedResults = myCustomerQuery.runPaged({
    pageSize: 10
});

// Fetch results using an iterator
var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
    var currentPage = resultPage.value;
    var currentPageRange = currentPage.pageRange;
    log.debug(currentPageRange.size);
    return true;
});

// Alternatively, fetch results using a loop
for (var i = 0; i < myPagedResults.pageRanges.length; i++) {
    var currentPage = myPagedResults.fetch(i);
    var currentPageRange = currentPage.pageRange;
    log.debug(currentPageRange.size);
}

PageRange.index

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describes the array index for this page range.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>number (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.PageRange</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>PageRange Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
})
```
```javascript
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
  myCustomerQuery.createColumn({
    fieldId: 'entityid'
  })
];

var myPagedResults = myCustomerQuery.runPaged({
  pageSize: 10
});

var iterator = myPagedResults.iterator();
iterator.each(function(resultPage) {
  var currentPage = resultPage.value;
  var currentPageRange = currentPage.pageRange;
  var theIndex = currentPageRange.index;
  return true;
});
...
// Add additional code
```

---

**PageRange.size**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Describes the number of query result rows in this page range.</td>
</tr>
<tr>
<td><strong>Type</strong></td>
<td>number (read-only)</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/query Module</td>
</tr>
<tr>
<td><strong>Parent Object</strong></td>
<td>query.PageRange</td>
</tr>
<tr>
<td><strong>Sibling Object Members</strong></td>
<td>PageRange Object Members</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2018.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
  myCustomerQuery.createColumn({
    fieldId: 'entityid'
  })
];
```
query.Query

| Object Description | The query.Query object encapsulates the query definition. To create a query with the N/query module:

1. Use the query.create(options) method to create your query definition (this object). The initial query definition uses one search type. For available search types, see query.Type.

2. After you create the initial query definition, use Query.autoJoin(options) to create your first join.

3. Then use Component.autoJoin(options) to create all subsequent joins.

The query definition always contains at least one query.Component object. The query.Component object encapsulates one component of the query definition. Each new component is created as a child to the previous component, and all components exist as children to the query definition.

You can think of a component as a building block; each new component builds on the previous component created. The last component created encapsulates the relationship between it and all of its parent components.

Queries with joins contain multiple components. The query definition contains a child query.Component object for each of the following:

- **The initial query definition**: The initial query.Component object is called the root component. It encapsulates the initial search type passed to query.create(options). The root component is automatically created with the initial query definition and is a child to the query.Query object. The Query.root property contains a reference to the root component.

- **The first join**: The second query.Component object is created with Query.autoJoin(options). It encapsulates the relationship between the initial query definition and the second search type. This relationship is determined by the join ID passed to Query.autoJoin(options). The second query.Component object is a child to the root component.

- **Each subsequent join**: The third query.Component object is created with Component.autoJoin(options). All subsequent joins are also created with Component.autoJoin(options). Each of these query.Component objects encapsulates the relationship between all previous search types and the new search type. This relationship is determined by the join ID passed to Component.autoJoin(options).

| Supported Script Types | Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.
## Query.and()

**Method Description**  
Creates a new condition (a `query.Condition` object) that corresponds to a logical conjunction (AND) of the arguments passed to the method. The arguments must be one or more `query.Condition` objects.

A condition narrows the query results. The `query.Condition` object acts in the same capacity as the `search.Filter` object in the `N/search Module`. The primary difference is that `query.Condition` objects can contain other `query.Condition` objects.

To create conditions:
- Use `Query.createCondition(options)` to create conditions for the initial query definition created with `query.create(options)`.
- Use `Component.createCondition(options)` to create conditions for the join relationships created with `Query.autoJoin(options)` and `Component.autoJoin(options)`.
- If you have multiple conditions, use them to create a new parent condition with the methods `Query.and()`, `Query.or()`, and `Query.not()`.
- Assign your parent condition to `Query.condition`. For an example, see Syntax.

**Returns**  
`query.Condition` object
**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition 1 — n</td>
<td>query.Condition</td>
<td>Required</td>
<td>One or more condition objects. There is no limit on the number of conditions you can specify.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});

var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});

var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});
```
search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();

**Query.autoJoin(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a join relationship.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use the method <code>query.create(options)</code> to create your initial query definition (<code>query.Query</code>). The initial query definition uses one search type. For available search types, see <code>query.Type</code>. After you create the initial query definition, use <code>Query.autoJoin(options)</code> to create your first join (<code>query.Component</code>). Then use <code>Component.autoJoin(options)</code> to create each subsequent join (<code>query.Component</code>).</td>
</tr>
</tbody>
</table>

**Important:** The `N/query` module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic Available Records.

<table>
<thead>
<tr>
<th>Returns</th>
<th><code>query.Component</code> object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td><code>N/query Module</code></td>
</tr>
<tr>
<td>Parent Object</td>
<td><code>query.Query</code></td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td><code>Query Object Members</code></td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.fieldId</code></td>
<td><code>string</code></td>
<td><code>required</code></td>
<td>The column type (field type) that joins the parent component to the new component. This value determines the columns on which the components are joined and the type of the newly joined component.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>------</td>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Obtain this value from the Records Browser:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Go to the parent component’s record type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Scroll until you see the Search Joins table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Locate the appropriate value in the Join ID column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATIONSHIP_ALREADY_USED</td>
<td>The specified join relationship already exists.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.autoJoin({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];

var results = search.runPaged({
    pageSize: 10
});
```

**Query.createColumn(options)**

**Method Description** This method creates a query result column based on the query.Query object.

The query.Column object is the equivalent of the search.Column object in the N/Search Module. The query.Column object describes the field types (columns) that are displayed from the query results.

To create columns:
Use `Query.createColumn(options)` to create conditions on the initial query definition created with `query.create(options)`. Use this method in one of two ways:

- Pass in an argument for the parameter `options.fieldId`.
- Pass in an argument for the parameter `options.formula`. If you use this option, you can also use the optional parameter `options.type`.

If needed, use `Component.createColumn(options)` to create conditions on the join relationships created with `Query.autoJoin(options)` and `Component.autoJoin(options)`.

Assign all created columns as array values to `Query.columns`. For an example, see `Syntax`.

**Note:** This method is a shortcut for the chained `Query.root` and `Component.createColumn(options)`. The `Query.root` property references the root component, which is a `query.Component` object.

**Returns**

- `query.Column object`

**Supported Script Types**

- Client and server-side scripts

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Governance**

- None

**Module**

- `N/query Module`

**Parent Object**

- `query.Query`

**Sibling Object Members**

- `Query Object Members`

**Since**

- 2018.1

---

**Parameters**

**Note:** The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| `options.fieldId`  | string    | required if `options.formula` is not used | The name of the query result column. This value sets the `Column.fieldId` property. Obtain this value from the Records Browser: 1. Go to the appropriate record type. 2. Scroll until you see the Search Columns table. 3. Locate the appropriate value in the Internal ID column. For more information on the Records Browser, see the help topic `Using the SuiteScript Records Browser`.
| `options.formula`  | string    | required if `options.fieldId` is not used | The formula used to create the query result column. This value sets the `Column.formula` property. For more information on formulas, see the help topics `Formulas in Search` and `SQL Expressions`. |
### Parameter Types

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>optional if options.formula is used</td>
<td>If you use the options.formula parameter, use this parameter to explicitly define the formula’s return type. Defining the formula’s return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the Column.type property. Use the appropriate query.ReturnType enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
<tr>
<td>options.aggregate</td>
<td>string</td>
<td>optional</td>
<td>Use this parameter to run an aggregate function on your query result column. An aggregate function performs a calculation on the column values and returns a single value. This value sets the Column.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
</tr>
<tr>
<td>options.consolidated</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether the query result column displays consolidated currency amounts. This value sets the Column.consolidated property. If you do not pass in an argument, the default value is set to false.</td>
</tr>
<tr>
<td>options.groupBy</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether the query results are grouped by this query result column. This value sets the Column.groupBy property. If you do not pass in an argument, the default value is set to false.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    })
];
```

---

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.
queryModule

SuiteScript 2.0 API Reference

<table>
<thead>
<tr>
<th>FieldId</th>
<th>'email'</th>
</tr>
</thead>
<tbody>
<tr>
<td>FieldId</td>
<td>'hiredate'</td>
</tr>
</tbody>
</table>

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();

Query.createCondition(options)

Method Description

This method creates a condition (query filter) based on the query.Query object. A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects.

To create conditions:

- Use Query.createCondition(options) to create conditions on the initial query definition created with query.create(options). Use this method in one of two ways:
  - Pass in arguments for the parameters options.fieldId, options.operator, and options.values. The combination of these arguments translates to <filter column><operator><field value> (for example, 'city' equals 'Boston').
  - Pass in an argument for the parameter options.formula. If you use this option, you can also use the optional parameter options.type.
- If needed, use Component.createCondition(options) to create conditions on the join relationships created with Query.autoJoin(options) and Component.autoJoin(options).
- If you have multiple conditions, use them to create a new nested condition with the methods Query.and(), Query.or(), and Query.not().
- Assign your simple or nested condition to Query.condition. For an example, see Syntax.

Note: This method is a shortcut for the chained Query.root and Component.createCondition(options): Query.root.createCondition(options). The Query.root property references the root component, which is a query.Component object.

Returns

query.Condition object

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.fieldId | string    | required if options.operator and options.values are used | The name of the condition. This value sets the `Condition.fieldId` property. Obtain this value from the Records Browser:  
1. Go to the appropriate record type.  
2. Scroll until you see the Search Filters table.  
3. Locate the appropriate value in the Internal ID column.  
   For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser. |
| options.operator| string    | required if options.fieldId and options.values are used | The operator used by the condition. This value sets the `Condition.operator` parameter. Use the appropriate `query.Operator` enum value to pass in your argument. This enum holds all the supported values for this parameter. |
| options/values  | string[]  | required if options.fieldId and options/operator are used | An array of string values. This value sets the `Condition.values` property.                                                                                                                                 |
| options.formula | string    | required if options.fieldId, options.operator, and options.values are not used | The formula used to create the condition. This value sets the `Condition.formula` property. For more information on formulas, see the help topics Formulas in Search and SQL Expressions. |
| options.type    | string    | optional if options.formula is used | If you use the `options.formula` parameter, use this parameter to explicitly define the formula's return type. Defining the formula's return type might be required if the return type cannot be determined correctly based on the specified formula. This value sets the `Condition.type` property.  
   Use the appropriate `query.ReturnType` enum value to pass in your argument. This enum holds all the supported values for this parameter. |
| options.aggregate| string    | optional             | Use this parameter to run an aggregate function on a condition. An aggregate function performs  

---

**Note:** The options parameter is a JavaScript object.
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a calculation on the condition values and returns a single value. This value sets the Condition.aggregate property. Use the appropriate query.Aggregate enum value to pass in your argument. This enum holds all the supported values for this parameter.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});
var salesrep = search.join({
    fieldId: 'salesrep'
});
var location = salesrep.join({
    fieldId: 'location'
});
var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});
search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);
var resultSet = search.run();
```

### Query.createSort(options)

**Method Description**

This method creates a sort based on the query.Query object. The query.Sort object describes a sort that is placed on a particular query result column.
To create a sort:

- Use `Search.createSort(options)` to create a sort based on the initial query definition created with `query.create(options)`.
- Use `Component.createSort(options)` to create a sort based on a join relationship created with `Query.autoJoin(options)` or `Component.autoJoin(options)`.
- Assign all created sorts as array values to `Query.sort`. For an example, see Syntax.

**Note:** This method is a shortcut for the chained `Query.root` and `Component.createSort(options); query.root.createSort(options)`. The `Query.root` property references the root component, which is a `query.Component` object.

Returns: query.Sort object

Supported Script Types: Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance: None

Module: N/query Module

Parent Object: query.Query

Sibling Object Members: Query Object Members

Since: 2018.1

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.column</td>
<td>query.Column</td>
<td>required</td>
<td>The query result column that you want to sort by. This value sets the Sort.column property.</td>
</tr>
<tr>
<td>options.ascending</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether the sort direction is ascending. This value sets the Sort.ascending property. The default value of this property is true, meaning that the sort direction is ascending. If you want the sort direction to be descending, set this property to false.</td>
</tr>
</tbody>
</table>
| options.caseSensitive| boolean             | optional            | Indicates whether the sort is case sensitive. This value sets the Sort.caseSensitive property. If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same. For example, the following list of items is sorted using a case-sensitive sort with a sort direction of ascending:
  - Banana |
Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending:

- apple
- Banana
- grapefruit
- kiwi
- Orange

The default value of this property is `false`.

**options.locale**  
**Type:** string  
**Required/Optional:** optional  
**Description:**

The locale to use for the sort. This value sets the `Sort.locale` property.

A locale represents a combination of language and region, and it can affect how certain values (such as strings) are sorted. For example, languages that share the same alphabet may sort characters differently. Use this property to ensure that query results are sorted using locale-specific rules.

Use the appropriate `query.SortLocale` enum value to pass in your argument. This enum holds all the supported values for this parameter.

**options.nullsLast**  
**Type:** boolean  
**Required/Optional:** optional  
**Description:**

Indicates whether query results with null values are listed at the end of the query results. This value sets the `Sort.nullsLast` property.

The default value of this property is the value of the `options.ascending` property. For example, if the `options.ascending` property is set to `true`, the `options.nullsLast` property is also set to `true`.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
  type: query.Type.CUSTOMER
});
var salesrep = search.join({
  fieldId: ‘salesrep’
});
search.columns = [
  search.createColumn({
```
```javascript
fieldId: 'entityid'
}),
search.createColumn({
    fieldId: 'id'
}),
salesrep.createColumn({
    fieldId: 'entityid'
}),
salesrep.createColumn({
    fieldId: 'email'
}),
salesrep.createColumn({
    fieldId: 'hiredate'
}),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

**Query.join(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a join relationship.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Important:</strong></td>
<td>This method is an alias to Query.autoJoin(options). Use Query.autoJoin(options) instead of this method to create simple joins. Use Query.joinFrom(options) and Query.joinTo(options) to create explicit directional joins.</td>
</tr>
</tbody>
</table>

Use the method `query.create(options)` to create your initial query definition (`query.Query`). The initial query definition uses one search type. For available search types, see `query.Type`.

After you create the initial query definition, use `query.join(options)` to create your first join (`query.Component`). Then use `Component.autoJoin(options)` to create each subsequent join (`query.Component`).

**Note:** This method is a shortcut for the chained `Query.root` and `Component.join(options): Query.root.join(options)`. The `Query.root` property references the root component, which is a `query.Component` object.

**Important:** The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic `Available Records`.

**Returns**

`query.Component`
Supported Script Types

Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/query Module

Parent Object

query.Query

Sibling Object Members

Query Object Members

Since

2018.1

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The column type (field type) that joins the parent component to the new component. This value determines the columns on which the components are joined and the type of the newly joined component.</td>
</tr>
</tbody>
</table>

Obtain this value from the Records Browser:

1. Go to the parent component's record type.
2. Scroll until you see the Search Joins table.
3. Locate the appropriate value in the Join ID column.

For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.join({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    fieldId: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    direction: query.SortDirection.ASCENDING
})];
```
Query.joinFrom(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates an explicit directional join relationship from another component to this component (an inverse join). This method sets the <code>Component.source</code> property on the returned <code>query.Component</code> object. Use the method <code>query.create(options)</code> to create your initial query definition (<code>query.Query</code>). The initial query definition uses one search type. For available search types, see <code>query.Type</code>. After you create the initial query definition, use this method to create your first join as an explicit directional join from another component to this component.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Note:</strong> This method is a shortcut for the chained <code>Query.root</code> and <code>Component.joinFrom(options)</code> methods. The <code>Query.root</code> property references the root component, which is a <code>query.Component</code> object.</td>
</tr>
<tr>
<td>Returns</td>
<td><code>query.Component</code> object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts For more information, see the help topic <code>SuiteScript 2.0 Script Types</code>.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td><code>N/query Module</code></td>
</tr>
<tr>
<td>Parent Object</td>
<td><code>query.Query</code></td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td><code>Query Object Members</code></td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The column type (field type) that joins the parent component to the new component.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>options.source</td>
<td>string</td>
<td>required</td>
<td>The search type of the component joined to this component. This value sets the Component.source property. This value can be described as the inverse relationship of this component, and it determines the source search type of the newly joined component.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATIONSHIP_ALREADY_USED</td>
<td>The specified join relationship already exists.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
var myEmployeeQuery = query.create(
  type: query.Type.EMPLOYEE
);)

var myTransactionJoin = myEmployeeQuery.joinFrom(
  fieldId: 'entity',
  source: 'transaction'
);

myEmployeeQuery.columns = [
  myEmployeeQuery.createColumn(
    fieldId: 'entityid'
  ),
  myTransactionJoin.createColumn(
    fieldId: 'entity'
  ),
  myTransactionJoin.createColumn(
    fieldId: 'daysoverdue'
  )
];
...
// Add additional code
```
Query.joinTo(options)

Method Description

Creates an explicit directional join relationship to another component from this component (a polymorphic join). This method sets the Component.target property on the returned query.Component object.

Use the method query.create(options) to create your initial query definition (query.Query). The initial query definition uses one search type. For available search types, see query.Type.

After you create the initial query definition, use this method to create your first join as an explicit directional join to another component from this component.

Note: This method is a shortcut for the chained Query.root and Component.joinTo(options); Query.root.autoJoin(options). The Query.root property references the root component, which is a query.Component object.

Important: The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic Available Records.

Returns
query.Component object

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/query Module

Parent Object
query.Query

Sibling Object Members
Query Object Members

Since
2018.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The column type (field type) that joins the parent component to the new component.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Obtain this value from the Records Browser:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Go to the parent component's record type.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Scroll until you see the Search Joins table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Locate the appropriate value in the Join ID column.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information on the Records Browser, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
<tr>
<td>options.target</td>
<td>string</td>
<td>required</td>
<td>The search type of the component joined to this component. This value sets the Component.target property.</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value can be described as the polymorphic relationship of this component, and it determines the target search type of the newly joined component.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>RELATIONSHIP_ALREADY_USED</td>
<td>The specified join relationship already exists.</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
  type: query.Type.TRANSACTION
});

var myEmployeeJoin = myTransactionQuery.joinTo({
  fieldId: 'createdby',
  target: 'employee'
});

myTransactionQuery.columns = [
  myTransactionQuery.createColumn({
    fieldId: 'entity'
  }),
  myEmployeeJoin.createColumn({
    fieldId: 'entityid'
  }),
  myEmployeeJoin.createColumn({
    fieldId: 'email'
  })
];
...
// Add additional code

### Query.run()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Executes the query and returns the query result set.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>query.ResultSet</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>10 Usage Units</td>
</tr>
</tbody>
</table>
Module: N/query Module

Parent Object: query.Query

Sibling Object Members: Query Object Members

Since: 2018.1

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
    search.createColumn({
        fieldId: 'id'
    }),
    salesrep.createColumn({
        fieldId: 'entityid'
    }),
    salesrep.createColumn({
        fieldId: 'email'
    }),
    salesrep.createColumn({
        fieldId: 'hiredate'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();
```

**Query.run.promise()**

**Method Description**: Executes the query asynchronously and returns the query result set.
### Query.runPaged()

**Method Description**
Executes the query and returns a set of paged results.

**Returns**
query.PagedData

**Supported Script Types**
Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**
10 Usage Units

**Module**
N/query Module

**Parent Object**
query.Query

**Sibling Object Members**
Query Object Members

**Since**
2018.1

---

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
var search = query.create({
    type: query.Type.TRANSACTION
});

var entity = search.join({
    fieldId: 'entity'
});

search.columns = [entity.createColumn({
    name: 'subsidiary'
})];

search.sort = [search.createSort({
    column: search.columns[0],
    ascending: false
})];
```
```javascript
var results = search.runPaged({
    pageSize: 10
});

// Use the count property to count the search results easily
var resultCount = search.runPaged({
    pageSize: 10
}).count;
```

**Query.runPaged.promise()**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Executes the query asynchronously and returns a set of paged results.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>query.PagedData</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>10 Usage Units</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Query</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Query Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

**Query.not()**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a new condition (a query.Condition object) that corresponds to a logical negation (NOT) of the argument passed to the method. The argument must be a query.Condition object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A condition narrows the query results. The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects.</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>query.Condition</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
</tbody>
</table>

To create conditions:

- Use `Query.createQuery(options)` to create conditions for the initial query definition created with `query.createQuery(options)`.  
- Use `Component.createQuery(options)` to create conditions for the join relationships created with `Query.autoJoin(options)` and `Component.autoJoin(options)`.  
- If you have multiple conditions, use them to create a new parent condition with the methods `Query.and()`, `Query.or()`, and `Query.not()`.  
- Assign your parent condition to `Query.condition`. For an example, see Syntax.
Module: N/query Module

Parent Object: query.Query

Sibling Object Members: Query Object Members

Since: 2018.1

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition</td>
<td>query.Condition</td>
<td>Required</td>
<td>One condition object.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

var location = salesrep.join({
    fieldId: 'location'
});

var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});

var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});

var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();
```
### Query.or()

**Method Description**

Creates a new condition (a `query.Condition` object) that corresponds to a logical disjunction (OR) of the arguments passed to the method. The arguments must be one or more `query.Condition` objects.

A condition narrows the query results. The `query.Condition` object acts in the same capacity as the `search.Filter` object in the `N/search Module`. The primary difference is that `query.Condition` objects can contain other `query.Condition` objects.

To create conditions:

- Use `Query.createCondition(options)` to create conditions for the initial query definition created with `query.create(options)`.
- Use `Component.createCondition(options)` to create conditions for the join relationships created with `Query.autoJoin(options)` and `Component.autoJoin(options)`.
- If you have multiple conditions, use them to create a new parent condition with the methods `Query.and()`, `Query.or()`, and `Query.not()`.
- Assign your parent condition to `query.condition`. For an example, see Syntax.

**Returns**

`query.Condition` object

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

`N/query Module`

**Parent Object**

`query.Query`

**Sibling Object Members**

Query Object Members

**Since**

2018.1

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>condition 1 — n</td>
<td><code>query.Condition</code></td>
<td>Required</td>
<td>One or more condition objects. There is no limit on the number of conditions you can specify.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
   type: query.Type.CUSTOMER
});

var salesrep = search.join({
   fieldId: 'salesrep'
});
```
var location = salesrep.join(
    fieldId: 'location'
);

var cond1 = search.createCondition(
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
);
var cond2 = search.createCondition(
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
);
var cond3 = salesrep.createCondition(
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
);

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();

---

### Query.child

#### Property Description
Holds a references to children of this component. The value of this property is an object of key/value pairs. Each key is the name of a child component. Each respective value is the corresponding `query.Component` object.

The object values are set with the execution of `Query.autoJoin(options)` and `Component.autoJoin(options)`. The order of the key/value pairs reflects the parent/child hierarchy.

<table>
<thead>
<tr>
<th>Type</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Query</td>
</tr>
<tr>
<td>Sibling Object</td>
<td>Query Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
```
```javascript
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

var myTaskJoin = myCustomerQuery.autoJoin({
    fieldId: 'task'
});

var theChild = myCustomerQuery.child;
...
// Add additional code
```

## Query.columns

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
</table>
|                     | Holds an array of result columns (query.Column objects) returned from the query. The query.Column object is the equivalent of the search.Column object in the `N/search Module`. The query.Column object describes a field type (column) that is returned from the query results. To create columns:
|                     | ■ Use `Query.createColumn(options)` to create conditions on the initial query definition created with `query.create(options)`.  
|                     | ■ Use `Component.createColumn(options)` to create conditions on the join relationships created with `Query.autoJoin(options)` and `Component.autoJoin(options)`.  
|                     | ■ Assign all created columns as array values to `Query.columns`. For an example, see `Syntax`. |

<table>
<thead>
<tr>
<th>Type</th>
<th><code>query.Column[]</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td><code>N/query Module</code></td>
</tr>
<tr>
<td>Parent Object</td>
<td><code>query.Query</code></td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td><code>Query Object Members</code></td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

## Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/query Module Script Samples`.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
```
Query.condition

Property Description
References the simple or nested condition (a query.Condition object) that narrows the query results.

The query.Condition object acts in the same capacity as the search.Filter object in the N/search Module. The primary difference is that query.Condition objects can contain other query.Condition objects.

To create conditions:
- Use Query.createCondition(options) to create conditions for the initial query definition created with query.create(options).
- Use Component.createCondition(options) to create conditions for the join relationships created with Query.autoJoin(options) and Component.autoJoin(options).
- If you have multiple conditions, use them to create a new nested condition with the methods Query.and(), Query.or(), and Query.not().
- Assign your simple or nested condition to Query.condition. For an example, see Syntax.

Type
query.Condition object

Module
N/query Module

Parent Object
query.Query

Sibling Object
Query Object Members

Since
2018.1

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'firstname'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    })
];
```
type: query.Type.CUSTOMER
});

var mySalesRepJoin = myCustomerQuery.join({
    fieldId: 'salesrep'
});

var myFirstCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});

var mySecondCondition = myCustomerQuery.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});

var myThirdCondition = myCustomerQuery.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

myCustomerQuery.condition = myCustomerQuery.and(
    myThirdCondition, myCustomerQuery.not(
        myCustomerQuery.or(myFirstCondition, mySecondCondition)
    )
);
...
// Add additional code

---

**Query.id**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Holds the ID of the query definition.</strong></td>
<td></td>
</tr>
</tbody>
</table>

This property has a value only for existing queries that are loaded using `query.load(options)`. If you create a query using `query.create(options)` but do not save it, this property is null.

⚠️ **Important:** The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. In the 2019.1 release, you can use the N/query module to load and delete existing queries, but you cannot save queries. You can save queries using the SuiteAnalytics Workbook interface. For more information, see the help topic [Navigating SuiteAnalytics Workbook](#).

<table>
<thead>
<tr>
<th>Type</th>
<th>number (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Query</td>
</tr>
<tr>
<td>Sibling Object</td>
<td>Query Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>
### Syntax

```javascript
// Add additional code
...
var myLoadedQuery = query.load({
   id: 'custworkbook237'  
});

var theId = myLoadedQuery.id;
...
// Add additional code
```

---

**Query.name**

| Property Description | Holds the name of the query definition.  
| --- | --- |
| This property has a value only for existing queries that are loaded using `query.load(options)`.  
| If you create a query using `query.create(options)` but do not save it, this property is null. |

---

**Important:** The `N/query` module lets you create and run queries using the SuiteAnalytics Workbook query engine. In the 2019.1 release, you can use the `N/query` module to load and delete existing queries, but you cannot save queries. You can save queries using the SuiteAnalytics Workbook interface. For more information, see the help topic [Navigating SuiteAnalytics Workbook](#).

---

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td><code>N/query Module</code></td>
</tr>
<tr>
<td>Parent Object</td>
<td><code>query.Query</code></td>
</tr>
<tr>
<td>Sibling Object</td>
<td><code>Query Object Members</code></td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
// Add additional code
...
var myLoadedQuery = query.load({
   id: 'custworkbook237'  
});

var theName = myLoadedQuery.name;
...
// Add additional code
```
Query.root

**Property Description**  References the root component of the query definition.

The initial `query.Component` object is called the root component. It encapsulates the initial search type passed to `query.create(options)`. The root component is automatically created with the `query.Query` object and is a child of the `query.Query` object.

**Type**  `query.Component` (read-only)

**Module**  `N/query Module`

**Parent Object**  `query.Query`

**Sibling Object Members**  `Query Object Members`

**Since** 2018.1

---

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/query Module Script Samples`.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

var theRoot = myCustomerQuery.root;
...
// Add additional code
```

---

Query.sort

**Property Description**  Holds an array of query result columns (`query.Column` objects) used for sorting.

This object encapsulates a sort based on the `query.Query` or `query.Component` object. The `query.Sort` object describes a sort that is placed on a particular query result column.

To create a sort:

- Use `Query.createSort(options)` to create a sort based on the initial query definition created with `query.create(options)`.
- Use `Component.createSort(options)` to create a sort based on a join relationship created with `Query.autoJoin(options)` or `Component.autoJoin(options)`.
- Assign all created sorts as array values to `query.sort`. For an example, see Syntax.

**Type**  `query.Sort[]`

**Module**  `N/query Module`

**Parent Object**  `query.Query`

**Sibling Object Members**  `Query Object Members`
Since 2018.1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

var mySalesRepJoin = myCustomerQuery.autoJoin({
    fieldId: 'salesrep'
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'firstname'
    }),
    mySalesRepJoin.createColumn({
        fieldId: 'email'
    })
];

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[1]
    }),
    mySalesRepJoin.createSort({
        column: myCustomerQuery.columns[0],
        ascending: false
    })
];
...
// Add additional code
```

Query.type

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query.type</td>
<td>Describes the initial search type of the query definition. This property is set during the execution of query.create(options).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Query</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Query Object Members</td>
</tr>
</tbody>
</table>
query.RelativeDate

Object Description
Encapsulates a relative date to use in query conditions.

Use `query.createDateRange(options)` to create this object. After you create this object, you can use it in the `values` parameter of `Query.createCondition(options)` or `Component.createCondition(options)`.

This object represents a specific moment in time, and you can use it to create query conditions using operators from the `query.Operator` enum, such as `query.Operator.AFTER`, `query.Operator.BEFORE`, and `query.Operator.WITHIN`. For more information about relative dates, see Relative Dates in the N/query Module.

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/query Module

Methods and Properties
RelativeDate Object Members

Since
2019.1

Syntax

```javascript
// Add additional code
...
var myEndDate = query.createDateRange({
  dateRangeId: query.DateRangeId.WEEKS_AGO,
  value: 2
});

var myComplexCondition = myQuery.createCondition({
```

```javascript
// Add additional code
...
```
fieldId: 'trandate',
    operator: query.Operator.WITHIN,
    values: [query.RelativeDateRange.THREE_FISCAL_YEARS_AGO.start, myEndDate]
});
...
// Add additional code

RelativeDate.dateRangeId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Holds the date range ID of the relative date. For relative date ranges that you create using <code>query createDateRange(options)</code>, the value of this property is set when that method is executed. For relative date ranges that are included in the <code>query.RelativeDateRange</code> enum, the value of this property is always available (for example, <code>query.RelativeDateRange.YESTERDAY.dateRangeId</code>). This property uses values from the <code>query.DateRangeId</code> enum.</td>
</tr>
</tbody>
</table>

| Type | string (read-only) |
| Module | N/query Module |
| Parent Object | query.RelativeDate |
| Sibling Object Members | RelativeDate Object Members |
| Since | 2019.1 |

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myRelativeDate = query.createDateRange({
  dateRangeId: query.DateRangeId.WEEKS_AGO,
  value: 2
});

var theDateRangeId = myRelativeDate.dateRangeId;
...
// Add additional code
```

RelativeDate.end

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>References the end of the relative date. For relative date ranges that you create using <code>query createDateRange(options)</code>, the value of this property is set when that method is executed. For relative date ranges that are included in the <code>query.RelativeDateRange</code> enum, the value of this property is always available (for example, <code>query.RelativeDateRange.YESTERDAY.end</code>).</td>
</tr>
</tbody>
</table>

| Type | Object (read-only) |
Module: N/query Module

Parent Object: query.RelativeDate

Sibling Object Members: RelativeDate Object Members

Since: 2019.1

Syntax

```
// Add additional code
...
var myRelativeDate = query.createDateRange({
    dateRangeId: query.DateRangeId.WEEKS_AGO,
    value: 2
});
var theDateRangeId = myRelativeDate.dateRangeId;
...
// Add additional code
```

RelativeDate.interval

**Property Description**

Describes the interval that the relative date represents.

For relative date ranges that you create using `query.createDateRange(options)`, the value of this property is set when that method is executed. For relative date ranges that are included in the `query.RelativeDateRange` enum, the value of this property is always available (for example, `query.RelativeDateRange.YESTERDAY.interval`).

**Important:** Do not use this property explicitly in your scripts. It is available so you can see the exact date interval that is used with the `query.Operator.WITHIN` and `query.Operator.WITHIN_NOT` operators in query conditions.

<table>
<thead>
<tr>
<th>Type</th>
<th>Object (read-only)</th>
</tr>
</thead>
</table>

Module: N/query Module

Parent Object: query.RelativeDate

Sibling Object Members: RelativeDate Object Members

Since: 2019.1

Syntax

```
// Add additional code
```
... var myRelativeDate = query.createDateRange({
    dateRangeId: query.DateRangeId.WEEKS_AGO,
    value: 2
});

var theInterval = myRelativeDate.interval;
...
// Add additional code

RelativeDate.start

<table>
<thead>
<tr>
<th>Property Description</th>
<th>References the start of the relative date.</th>
</tr>
</thead>
<tbody>
<tr>
<td>For relative date ranges that you create using query.createDateRange(options), the value of this property is set when that method is executed. For relative date ranges that are included in the query.RelativeDateRange enum, the value of this property is always available (for example, query.RelativeDateRange.YESTERDAY.start).</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>Object (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.RelativeDate</td>
</tr>
<tr>
<td>Sibling Object</td>
<td>RelativeDate Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myRelativeDate = query.createDateRange({
    dateRangeId: query.DateRangeId.WEEKS_AGO,
    value: 2
});

var theStart = myRelativeDate.start;
...
// Add additional code
```

RelativeDate.value

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds the value of the relative date range.</th>
</tr>
</thead>
<tbody>
<tr>
<td>For relative date ranges that you create using query.createDateRange(options), the value of this property is set when that method is executed. For relative date ranges that are included in the query.RelativeDateRange enum, the value of this property is undefined (for example, query.RelativeDateRange.YESTERDAY.value is undefined).</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>number (read-only)</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.RelativeDate</td>
</tr>
<tr>
<td>Sibling Object</td>
<td>RelativeDate Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myRelativeDate = query.createDateRange({
  dateRangeId: query.DateRangeId.WEEKS_AGO,
  value: 2
});

var theValue = myRelativeDate.value;
...
// Add additional code
```

**query.Result**

Object Description Encapsulates a single row of the result set (query.ResultSet).

Supported Script Types Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module N/query Module

Methods and Properties Result Object Members

Since 2018.1

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});
```
myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'firstname'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var queryResultSet = myCustomerQuery.run();

// Fetch results using an iterator
var iterator = queryResultSet.iterator();
iterator.each(function(result) {
    var currentResult = result.value;
    log.debug(currentResult);
    return true;
});

// Alternatively, fetch results using a loop
var queryResults = queryResultSet.results;
for (var i = 0; i < queryResults.length; i++) {
    var currentResult = queryResults[i];
    log.debug(currentResult);
}
...
// Add additional code

---

### Result.columns

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds an array of query return column references. These array values are equivalent to the array values in ResultSet.columns.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>query.Column[] (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.Result</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Result Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

// Add additional code
... var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    })
];

var queryResultSet = myCustomerQuery.run();

var queryResults = queryResultSet.results;
var myFirstResult = queryResults[0];
var theColumns = myFirstResult.columns;
...
// Add additional code

### Result.values

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result.values</td>
<td>Describes the result values. Value types correspond to the <code>ResultSet.types</code> property. Array values correspond to the array values for <code>ResultSet.columns</code> and <code>ResultSet.values</code>.</td>
</tr>
</tbody>
</table>

**Type**

Array<string | number | boolean | null> (read-only)

**Module**

N/query Module

**Parent Object**

query.Result

**Sibling Object Members**

Result Object Members

**Since**

2018.1

### Syntax

⚠️ **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];
```
var queryResultSet = myCustomerQuery.run();

var queryResults = queryResultSet.results;
var myFirstResult = queryResults[0];
var theValues = myFirstResult.values;
...
// Add additional code

query.ResultSet

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the set of results returned by the query. Use Query.run() or Query.run.promise() to create this object.</th>
</tr>
</thead>
</table>
| Supported Script Types | Client and server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Module | N/query Module |
| Methods and Properties | ResultSet Object Members |
| Since | 2018.1 |

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var resultSet = search.run();
var results = resultSet.results;
for (var i = results.length - 1; i >= 0; i--)
    log.debug(results[i].values);
```

ResultSet.iterator()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Standard SuiteScript 2.0 object for iterating through results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Iterator object</td>
</tr>
</tbody>
</table>
| Supported Script Types | Client and server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance | None |
| Module | N/query Module |
| Parent Object | query.ResultSet |
| Sibling Object Members | ResultSet Object Members |
| Since | 2018.1 |
Syntax

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
   type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
   myCustomerQuery.createColumn({
      fieldId: 'entityid'
   }),
   myCustomerQuery.createColumn({
      fieldId: 'firstname'
   }),
   myCustomerQuery.createColumn({
      fieldId: 'email'
   })
];

var queryResultSet = myCustomerQuery.run();

// Fetch results using an iterator
var iterator = queryResultSet.iterator();
iterator.each(function(result) {
   var currentResult = result.value;
   log.debug(currentResult);
   return true;
});

// Alternatively, fetch results using a loop
var queryResults = queryResultSet.results;
for (var i = 0; i < queryResults.length; i++) {
   var currentResult = queryResults[i];
   log.debug(currentResult);
}
...
// Add additional code
```

ResultSet.columns

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holds an array of query return column references. The <code>ResultSet.columns</code> array values correspond with the <code>ResultSet.types</code> array values.</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>query.Column[] (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.ResultSet</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>ResultSet Object Members</td>
</tr>
</tbody>
</table>
N/query Module

Since 2018.1

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
  myCustomerQuery.createColumn({
    fieldId: 'entityid'
  }),
  myCustomerQuery.createColumn({
    fieldId: 'email'
  })
];

var queryResultSet = myCustomerQuery.run();

var theColumns = queryResultSet.columns;
...
// Add additional code
```

ResultSet.results

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds an array of query.Result objects.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>query.Result[] (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.ResultSet</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>ResultSet Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});
```
myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var queryResultSet = myCustomerQuery.run();

var theResults = queryResultSet.results;
...
// Add additional code

var myCustomerQuery = query.create({
    type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
    myCustomerQuery.createColumn({
        fieldId: 'entityid'
    }),
    myCustomerQuery.createColumn({
        fieldId: 'email'
    })
];

var queryResultSet = myCustomerQuery.run();

var theTypes = queryResultSet.types;

ResultSet.types

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Holds an array of the return types for ResultSet.results. The ResultSet.types array values correspond with the ResultSet.columns array values.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string[] (read-only)</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Parent Object</td>
<td>query.ResultSet</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>ResultSet Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.
// Add additional code

query.Sort

Object Description
Encapsulates a sort based on the query.Query or query.Component object. The query.Sort object describes a sort that is placed on a particular query result column.

To create a sort:
- Use Query.createSort(options) to create a sort based on the initial query definition created with query.create(options).
- Use Component.createSort(options) to create a sort based on a join relationship created with Query.autojoin(options) or Component.autojoin(options).
- Assign all created sorts as array values to Query.sort. For an example, see Syntax.

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/query Module

Methods and Properties
Sort Object Members

Since
2018.1

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create(
    type: query.Type.CUSTOMER);

var salesrep = search.join(
    fieldId: 'salesrep');

search.columns = [
    search.createColumn(
        fieldId: 'entityid'),
    search.createColumn(
        fieldId: 'id'),
    salesrep.createColumn(
        fieldId: 'entityid'),
    salesrep.createColumn(
        fieldId: 'email'),
    salesrep.createColumn(
        fieldId: 'hiredate')
];
```
Sort.ascending

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether the sort direction is ascending.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property is set during the execution of Query.createSort(options) and Component.createSort(options).</td>
</tr>
<tr>
<td></td>
<td>The default value of this property is true, meaning that the sort direction is ascending.</td>
</tr>
<tr>
<td></td>
<td>If you want the sort direction to be descending, set this property to false.</td>
</tr>
</tbody>
</table>

**Type**

boolean

**Module**

N/query Module

**Parent Object**

query.Sort

**Sibling Object Members**

Sort Object Members

**Since**

2018.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
  type: query.Type.CUSTOMER
});

search.columns = [
  search.createColumn({
    fieldId: 'entityid'
  })
];

search.sort = [
  search.createSort({
    column: search.columns[1],
  })],
  salesrep.createSort({
    column: salesrep.columns[0],
    ascending: false
  })
];

var resultSet = search.run();
```
nullsLast: false
});
);

var resultSet = search.run();

Sort.caseSensitive

### Property Description
Indicates whether the sort is case sensitive.

This property is set during the execution of `Query.createSort(options)` and `Component.createSort(options)`.

If a sort is case sensitive (and the sort direction is ascending), rows with column values that start with uppercase letters are listed before rows with column values that start with lowercase letters. If a sort is not case sensitive, uppercase and lowercase letters are treated the same. For example, the following list of items is sorted using a case-sensitive sort with a sort direction of ascending:

- Banana
- Orange
- apple
- grapefruit
- kiwi

Here is the same list of items sorted using a regular (not case-sensitive) sort with a sort direction of ascending:

- apple
- Banana
- grapefruit
- kiwi
- Orange

The default value of this property is `false`.

### Type
boolean

<table>
<thead>
<tr>
<th>Module</th>
<th>N/query Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Object</td>
<td>query.Sort</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Sort Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
  type: query.Type.CUSTOMER
});
```
search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
N/query Module

locale: query.SortLocale.EN_CA,
nullsLast: false
})
];
var resultSet = search.run();

Sort.locale
Property
Description

The locale to use for the sort.
This property uses values from the query.SortLocale enum. This property is set during the
execution of Query.createSort(options) and Component.createSort(options).
A locale represents a combination of language and region, and it can affect how certain
values (such as strings) are sorted. For example, languages that share the same alphabet
may sort characters differently. Use this property to ensure that query results are sorted
using locale-specific rules.

Type

string

Module

N/query Module

Parent Object

query.Sort

Sibling Object
Members

Sort Object Members

Since

2018.2

Syntax
Important: The following code snippet shows the syntax for this member. It is not a

functional example. For a complete script example, see N/query Module Script Samples.
var search = query.create({
type: query.Type.CUSTOMER
});
search.columns = [
search.createColumn({
fieldId: 'entityid'
})
];
search.sort = [
search.createSort({
column: search.columns[0],
ascending: false,
caseSensitive: true,
locale: query.SortLocale.EN_CA,
nullsLast: false
})
];
var resultSet = search.run();

SuiteScript 2.0 API Reference

481


Sort.nullsLast

**Property Description**
Indicates whether query results with null values are listed at the end of the query results.

This property is set during the execution of `Query.createSort(options)` and `Component.createSort(options)`.

The default value of this property is the value of the `Sort.ascend`ing property. For example, if the `Sort.ascend`ing property is set to `true`, the `Sort.nullsLast` property is also set to `true`.

**Type**
boolean

**Module**
N/query Module

**Parent Object**
query.Sort

**Sibling Object Members**
Sort Object Members

**Since**
2018.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    })
];

search.sort = [
    search.createSort({
        column: search.columns[0],
        ascending: false,
        caseSensitive: true,
        locale: query.SortLocale.EN_CA,
        nullsLast: false
    })
];

var resultSet = search.run();
```

**query.create(options)**

**Method Description**
Creates a `query.Query` object.

Use this method to create your initial query definition. The initial query definition uses one search type. For available search types, see `query.Type`.
After you create the initial query definition, use `Query.autoJoin(options)` to create your first join. Then use `Component.autoJoin(options)` to create all subsequent joins.

For more information about creating queries, see Scripting with the N/query Module.

**Returns**

query.Query object

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/query Module

**Sibling Module Members**

N/query Module Members

**Since**

2018.1

---

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The search type that you want to use for the initial query definition. Use the <code>query.Type</code> enum to set this value (for an example, see the help topic Syntax). When you execute <code>query.create(options)</code>, the <code>Query.type</code> property is set based on this value.</td>
</tr>
</tbody>
</table>

**Important:** The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic Available Records.

---

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});

search.columns = [
    search.createColumn({
        fieldId: 'entityid'
    }),
];
```
search.createColumn({
    fieldId: 'id'
}),
salesrep.createColumn({
    fieldId: 'entityid'
}),
salesrep.createColumn({
    fieldId: 'email'
}),
salesrep.createColumn({
    fieldId: 'hiredate'
}),
];

search.sort = [
    search.createSort({
        column: search.columns[1]
    }),
    salesrep.createSort({
        column: salesrep.columns[0],
        ascending: false
    })
];

var resultSet = search.run();

query.createDateRange(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a query.RelativeDate object that represents a range of dates relative to the current date.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use this method to create a query.RelativeDate object to use as part of a query condition. After you create a query.RelativeDate object, you can use it directly in the values parameter of Query.createCondition(options) or Component.createCondition(options).</td>
</tr>
<tr>
<td></td>
<td>When you call this method, the options.dateRangeId parameter determines the date range that is created. The options.dateRangeId parameter uses values from the query.DateRangeId enum, and these values describe potential date ranges relative to the current date. Use them in conjunction with the options.value parameter to create a relative date range. For example, to create a date range that includes dates from three weeks ago until the current date, call query.createDateRange(options) with an options.dateRangeId value of query.DateRangeId.WEEKS_AGO and an options.value value of 3. To create a date range that includes dates from the current date until three weeks from now, call query.createDateRange(options) with an options.dateRangeId value of query.DateRangeId.WEEKS_FROM_NOW and an options.value value of 3.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>query.RelativeDate object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module Members</td>
<td>N/query Module Members</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.dateRangeId</td>
<td>string</td>
<td>required</td>
<td>The ID of the date range to create. Use the <code>query.DateRangeId</code> enum to set this value.</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>required</td>
<td>The value to use to create the date range. This value depends on the value that you specify for <code>options.dateRangeId</code>. For example, to create a date range that includes dates from five days ago until the current date, use an <code>options.value</code> value of 5 and an <code>options.dateRangeId</code> value of <code>query.DateRangeId.DAYS_AGO</code>.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
  type: query.Type.TRANSACTION
});

var myDateRange = query.createDateRange({
  dateRangeId: query.DateRangeId.DAYS_AGO,
  value: 5
});

myTransactionQuery.condition = myTransactionQuery.createCondition({
  fieldId: 'trandate',
  operator: query.Operator.WITHIN,
  values: myDateRange
});

// Add additional code
```

`query.delete(options)`

**Method Description** Deletes an existing query.

Use this method to delete a query definition that was previously created using the SuiteAnalytics Workbook UI. After the query is deleted, it is no longer available and cannot be modified or executed.
Important: The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. In the 2019.1 release, you can use the N/query module to load and delete existing queries, but you cannot save queries. You can save queries using the SuiteAnalytics Workbook interface. For more information, see the help topic Navigating SuiteAnalytics Workbook.

Returns: void

Supported Script Types: Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance: 5 Usage Units

Module: N/query Module

Sibling Module Members: N/query Module Members

Since: 2018.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The script ID of the query to delete.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNABLE_TO_DELETE_QUERY</td>
<td>A query with the specified ID cannot be deleted because the query does not exist or you do not have permission to delete it.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
var deletedSearch = query.delete({
    id: 'custworkbook237'
});
```

query.load(options)

Method Description: Loads an existing query as a query.Query object.

Use this method to load a query definition that was previously created using the SuiteAnalytics Workbook UI. After the query is loaded, you can modify the query definition (for example, by setting additional property values), join the query definition with other
search types, and execute the query in the same way as queries that you create using `query.create(options)`.

**Important:** The N/query module lets you create and run queries using the SuiteAnalytics Workbook query engine. In the 2019.1 release, you can use the N/query module to load and delete existing queries, but you cannot save queries. You can save queries using the SuiteAnalytics Workbook interface. For more information, see the help topic Navigating SuiteAnalytics Workbook.

### Returns
query.Query object

### Supported Script Types
Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types. 

### Governance
5 Usage Units

### Module
N/query Module

### Sibling Module Members
N/query Module Members

### Since
2018.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The script ID of the query to load.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNABLE_TO_LOAD_QUERY</td>
<td>A query with the specified ID cannot be loaded because the query does not exist or you do not have permission to load it.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myLoadedQuery = query.load({
    id: 'custworkbook237'
});

var mySalesRepJoin = myLoadedQuery.setAutoJoin({
    fieldId: 'salesrep'
});
```
var results = myLoadedQuery.run();
...
// add additional code

query.Aggregate

Enum Description
Holds the string values for aggregate functions supported with the N/query Module. An aggregate function performs a calculation on the column or condition values and returns a single value.

Each value in this enum (except MEDIAN) has two variants: distinct (using the _DISTINCT suffix) and nondistinct (using no suffix). The variant determines whether the aggregate function operates on all instances of duplicate values or on just a single instance of the value. For example, consider a situation in which the MAXIMUM aggregate function is used to determine the maximum of a set of values. When using the distinct variant (MAXIMUM_DISTINCT), the aggregate function considers each instance of duplicate values. So if the set of values includes three distinct values that are all equal and all represent the maximum value in the set, the aggregate function lists all three instances. When using the nondistinct variant (MAXIMUM), only one instance of the maximum value is listed, regardless of the number of instances of that maximum value in the set.

This enum is used to pass the aggregate function argument to Component.createColumn(options), Component.createCondition(options), Query.createColumn(options), and Query.createCondition(options).

Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module Members</td>
<td>N/query Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVERAGE</td>
<td>Calculates the average value.</td>
</tr>
<tr>
<td>AVERAGE_DISTINCT</td>
<td>Calculates the average distinct value.</td>
</tr>
<tr>
<td>COUNT</td>
<td>Counts the number of results.</td>
</tr>
<tr>
<td>COUNT_DISTINCT</td>
<td>Counts the number of distinct results.</td>
</tr>
<tr>
<td>MAXIMUM</td>
<td>Determines the maximum value. If the values are dates, the most recent date is determined.</td>
</tr>
<tr>
<td>MAXIMUM_DISTINCT</td>
<td>Determines the maximum distinct value. If the values are dates, the most recent date is determined.</td>
</tr>
<tr>
<td>MEDIAN</td>
<td>Calculates the median value.</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINIMUM</td>
<td>Determines the minimum value. If the values are dates, the earliest date is determined.</td>
</tr>
<tr>
<td>MINIMUM_DISTINCT</td>
<td>Determines the minimum distinct value. If the values are dates, the earliest date is determined.</td>
</tr>
<tr>
<td>SUM</td>
<td>Adds all values.</td>
</tr>
<tr>
<td>SUM_DISTINCT</td>
<td>Adds all distinct values.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
   type: query.Type.TRANSACTION
});

var myAggColumn = myTransactionQuery.createColumn({
   fieldId: 'amount',
   aggregate: query.Aggregate.AVERAGE
});

myTransactionQuery.columns = [myAggColumn];
...
// Add additional code
```

### query.DateRangeId

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported date range codes in relative dates.</th>
</tr>
</thead>
</table>

This enum is used to pass the date range argument to `query.createDateRange(options)`. It is also used as the value of the `RelativeDate.dateRangeId` property. When `query.createDateRange(options)` is called, the enum value that you specify is set as the value of the `RelativeDate.dateRangeId` property.

When creating a date range using `query.createDateRange(options)`, use the values in this enum to specify a range of dates relative to the current date. For example, to create a date range for a certain number of days before the current day, use the `DateRangeId.DAYS_AGO` enum value. To create a date range for a certain number of months after the current day, use the `DateRangeId.MONTHS_FROM_NOW` enum value.

The values in this enum might look similar to the values in the `query.RelativeDateRange` enum, but each enum is used for a different purpose:

- Use the `query.DateRangeId` enum to create a `query.RelativeDate` object using `query.createDateRange(options)`. After you create this object, you can use it in query conditions that you create using `Query.createCondition(options)` or `Component.createCondition(options)`.
- Use the `query.RelativeDateRange` enum directly in query conditions that you create using `query.createCondition(options)` or `Component.createCondition(options)`. Each value in the `query.RelativeDateRange` enum maps to a `query.RelativeDate` object, and
you can use these values in the values parameter of `Query.createCondition(options)` or `Component.createCondition(options)`.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module</td>
<td>N/query Module Members</td>
</tr>
<tr>
<td>Members</td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Sets RelativeDate.dateRangeId Property To</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAYS_AGO</td>
<td>dago</td>
</tr>
<tr>
<td>DAYS_FROM_NOW</td>
<td>dfn</td>
</tr>
<tr>
<td>HOURS_AGO</td>
<td>hago</td>
</tr>
<tr>
<td>HOURS_FROM_NOW</td>
<td>hfn</td>
</tr>
<tr>
<td>MINUTES_AGO</td>
<td>nago</td>
</tr>
<tr>
<td>MINUTES_FROM_NOW</td>
<td>nfn</td>
</tr>
<tr>
<td>MONTHS_AGO</td>
<td>mago</td>
</tr>
<tr>
<td>MONTHS_FROM_NOW</td>
<td>mfn</td>
</tr>
<tr>
<td>QUARTERS_AGO</td>
<td>qago</td>
</tr>
<tr>
<td>QUARTERS_FROM_NOW</td>
<td>qfn</td>
</tr>
<tr>
<td>SECONDS_AGO</td>
<td>sago</td>
</tr>
<tr>
<td>SECONDS_FROM_NOW</td>
<td>sfn</td>
</tr>
<tr>
<td>WEEKS_AGO</td>
<td>wago</td>
</tr>
<tr>
<td>WEEKS_FROM_NOW</td>
<td>wfn</td>
</tr>
<tr>
<td>YEARS_AGO</td>
<td>yago</td>
</tr>
<tr>
<td>YEARS_FROM_NOW</td>
<td>yfn</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
```
var dateRange = query.createDateRange({
  dateRangeId: query.DateRangeId.DAYS_AGO,
  value: 2
});
...
// Add additional code

query.Operator

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for operators supported with the N/query Module. This enum is used to pass the operator argument to Query.createCondition(options) and Component.createCondition(options).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module Members</td>
<td>N/query Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

Values

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFTER</td>
</tr>
<tr>
<td>AFTER_NOT</td>
</tr>
<tr>
<td>ANY_OF</td>
</tr>
<tr>
<td>ANY_OF_NOT</td>
</tr>
<tr>
<td>BEFORE</td>
</tr>
<tr>
<td>BEFORE_NOT</td>
</tr>
<tr>
<td>BETWEEN</td>
</tr>
<tr>
<td>BETWEEN_NOT</td>
</tr>
<tr>
<td>CONTAIN</td>
</tr>
<tr>
<td>CONTAIN_NOT</td>
</tr>
<tr>
<td>EMPTY</td>
</tr>
<tr>
<td>EMPTY_NOT</td>
</tr>
<tr>
<td>ENDWITH</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>ENDWITH_NOT</td>
</tr>
<tr>
<td>EQUAL</td>
</tr>
<tr>
<td>EQUAL_NOT</td>
</tr>
<tr>
<td>GREATER</td>
</tr>
<tr>
<td>GREATER_NOT</td>
</tr>
<tr>
<td>GREATER_OR_EQUAL</td>
</tr>
<tr>
<td>GREATER_OR_EQUAL_NOT</td>
</tr>
<tr>
<td>IS</td>
</tr>
<tr>
<td>IS_NOT</td>
</tr>
<tr>
<td>LESS</td>
</tr>
<tr>
<td>LESS_NOT</td>
</tr>
<tr>
<td>LESS_OR_EQUAL</td>
</tr>
<tr>
<td>LESS_OR_EQUAL_NOT</td>
</tr>
<tr>
<td>ON</td>
</tr>
<tr>
<td>ON_NOT</td>
</tr>
<tr>
<td>ON_OR_AFTER</td>
</tr>
<tr>
<td>ON_OR_AFTER_NOT</td>
</tr>
<tr>
<td>ON_OR_BEFORE</td>
</tr>
<tr>
<td>ON_OR_BEFORE_NOT</td>
</tr>
<tr>
<td>START_WITH</td>
</tr>
<tr>
<td>START_WITH_NOT</td>
</tr>
<tr>
<td>WITHIN</td>
</tr>
<tr>
<td>WITHIN_NOT</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
var search = query.create({
    type: query.Type.CUSTOMER
});

var salesrep = search.join({
    fieldId: 'salesrep'
});
```
var cond1 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 107
});
var cond2 = search.createCondition({
    fieldId: 'id',
    operator: query.Operator.EQUAL,
    values: 2647
});
var cond3 = salesrep.createCondition({
    fieldId: 'email',
    operator: query.Operator.START_WITH_NOT,
    values: 'foo'
});

search.condition = search.and(
    cond3, search.not(
        search.or(cond1, cond2)
    )
);

var resultSet = search.run();

---

**query.RelativeDateRange**

<table>
<thead>
<tr>
<th>Enum Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Holds <code>query.RelativeDate</code> object values for supported date ranges in relative dates.</td>
</tr>
</tbody>
</table>

This enum is used to pass the `values` argument to `Query.createCondition(options)` and `Component.createCondition(options)`. It is also used as the value of the `RelativeDate.value` property. Each value in this enum represents a date range. When `Query.createCondition(options)` or `Component.createCondition(options)` is called with a `query.RelativeDate` object as the `values` argument, this object is set as the value of the `RelativeDate.value` property.

When creating a condition using `Query.createCondition(options)` or `Component.createCondition(options)`, use the values in this enum (in conjunction with values in the `query.Operator` enum) to specify a range of dates relative to the current date. For example, to create a condition to match dates that occur before the current day, use the `RelativeDateRange.TODAY` enum value and the `Operator.BEFORE` enum value. To create a condition to match dates that occur after last year, use the `RelativeDateRange.LAST_YEAR` enum value and the `Operator.AFTER` enum value. For more information about relative dates, see Relative Dates in the N/query Module.

The values in this enum might look similar to the values in the `query.DateRangeId` enum, but each enum is used for a different purpose:

- Use the `query.DateRangeId` enum to create a `query.RelativeDate` object using `query.createDateRange(options)`. After you create this object, you can use it in query conditions that you create using `Query.createCondition(options)` or `Component.createCondition(options).
- Use the `query.RelativeDateRange` enum directly in query conditions that you create using `Query.createCondition(options)` or `Component.createCondition(options)`. Each value in the `query.RelativeDateRange` enum represents a date range, and you can use these values in the `values` parameter of `Query.createCondition(options)` or `Component.createCondition(options).`
**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module Members</td>
<td>N/query Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Values

<table>
<thead>
<tr>
<th>Value</th>
<th>RelativeDate.dateRangeId Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>FISCAL_HALF_BEFORE_LAST</td>
<td>FHBL</td>
</tr>
<tr>
<td>FISCAL_HALF_BEFORE_LAST_TO_DATE</td>
<td>FHBLTD</td>
</tr>
<tr>
<td>FISCAL_QUARTER_BEFORE_LAST</td>
<td>FQBL</td>
</tr>
<tr>
<td>FISCAL_QUARTER_BEFORE_LAST_TO_DATE</td>
<td>FQBLTD</td>
</tr>
<tr>
<td>FISCAL_YEAR_BEFORE_LAST</td>
<td>FYBL</td>
</tr>
<tr>
<td>FISCAL_YEAR_BEFORE_LAST_TO_DATE</td>
<td>FYBLTD</td>
</tr>
<tr>
<td>FIVE_DAYS_AGO</td>
<td>DAGO5</td>
</tr>
<tr>
<td>FIVE_DAYS_FROM_NOW</td>
<td>DFNS</td>
</tr>
<tr>
<td>FOUR_DAYS_AGO</td>
<td>DAGO4</td>
</tr>
<tr>
<td>FOUR_DAYS_FROM_NOW</td>
<td>DFN4</td>
</tr>
<tr>
<td>FOUR WEEKS_STARTING_THIS_WEEK</td>
<td>TWN3W</td>
</tr>
<tr>
<td>LAST_BUSINESS_WEEK</td>
<td>LBW</td>
</tr>
<tr>
<td>LAST_FISCAL_HALF</td>
<td>LFH</td>
</tr>
<tr>
<td>LAST_FISCAL_HALF_ONE_FISCAL_YEAR_AGO</td>
<td>LFHLFY</td>
</tr>
<tr>
<td>LAST_FISCAL_HALF_TO_DATE</td>
<td>LFHTD</td>
</tr>
<tr>
<td>LAST_FISCAL_QUARTER</td>
<td>LFQ</td>
</tr>
<tr>
<td>LAST_FISCAL_QUARTER_ONE_FISCAL_YEAR_AGO</td>
<td>LFQLFY</td>
</tr>
<tr>
<td>LAST_FISCAL_QUARTER_TO_DATE</td>
<td>LFQTD</td>
</tr>
<tr>
<td>LAST_FISCAL_QUARTER_TWO_FISCAL_YEARS_AGO</td>
<td>LFQFYBL</td>
</tr>
<tr>
<td>LAST_FISCAL_YEAR</td>
<td>LFY</td>
</tr>
<tr>
<td>LAST_FISCAL_YEAR_TO_DATE</td>
<td>LFYTD</td>
</tr>
<tr>
<td>LAST_MONTH</td>
<td>LM</td>
</tr>
<tr>
<td>Value</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>LAST_MONTH_ONE_FISCAL_QUARTER_AGO</td>
<td>LMLFQ</td>
</tr>
<tr>
<td>LAST_MONTH_ONE_FISCAL_YEAR_AGO</td>
<td>LMLFY</td>
</tr>
<tr>
<td>LAST_MONTH_TO_DATE</td>
<td>LMTD</td>
</tr>
<tr>
<td>LAST_MONTH_TWO_FISCAL_QUARTERS_AGO</td>
<td>LMFQBL</td>
</tr>
<tr>
<td>LAST_MONTH_TWO_FISCAL_YEARS_AGO</td>
<td>LMFYBL</td>
</tr>
<tr>
<td>LAST_ROLLING_HALF</td>
<td>LRH</td>
</tr>
<tr>
<td>LAST_ROLLING_QUARTER</td>
<td>LRQ</td>
</tr>
<tr>
<td>LAST_ROLLING_YEAR</td>
<td>LRY</td>
</tr>
<tr>
<td>LAST_WEEK</td>
<td>LW</td>
</tr>
<tr>
<td>LAST_WEEK_TO_DATE</td>
<td>LWTD</td>
</tr>
<tr>
<td>LAST_YEAR</td>
<td>LY</td>
</tr>
<tr>
<td>LAST_YEAR_TO_DATE</td>
<td>LYTD</td>
</tr>
<tr>
<td>MONTH_AFTER_NEXT</td>
<td>MAN</td>
</tr>
<tr>
<td>MONTH_AFTER_NEXT_TO_DATE</td>
<td>MANTD</td>
</tr>
<tr>
<td>MONTH_BEFORE_LAST</td>
<td>MBL</td>
</tr>
<tr>
<td>MONTH_BEFORE_LAST_TO_DATE</td>
<td>MBLTD</td>
</tr>
<tr>
<td>NEXT_BUSINESS_WEEK</td>
<td>NBW</td>
</tr>
<tr>
<td>NEXT_FISCAL_HALF</td>
<td>NFH</td>
</tr>
<tr>
<td>NEXT_FISCAL_QUARTER</td>
<td>NFQ</td>
</tr>
<tr>
<td>NEXT_FISCAL_YEAR</td>
<td>NPY</td>
</tr>
<tr>
<td>NEXT_FOUR WEEKS</td>
<td>N4W</td>
</tr>
<tr>
<td>NEXT_MONTH</td>
<td>NM</td>
</tr>
<tr>
<td>NEXT_ONE_HALF</td>
<td>NOH</td>
</tr>
<tr>
<td>NEXT_ONE_MONTH</td>
<td>NOM</td>
</tr>
<tr>
<td>NEXT_ONE_QUARTER</td>
<td>NOQ</td>
</tr>
<tr>
<td>NEXT_ONE_WEEK</td>
<td>NOW</td>
</tr>
<tr>
<td>NEXT_ONE_YEAR</td>
<td>NOY</td>
</tr>
<tr>
<td>NEXT_WEEK</td>
<td>NW</td>
</tr>
<tr>
<td>NINETY_DAYS_AGO</td>
<td>DAGO90</td>
</tr>
<tr>
<td>NINETY_DAYS_FROM_NOW</td>
<td>DFN90</td>
</tr>
<tr>
<td>ONE_YEAR_BEFORE_LAST</td>
<td>OYBL</td>
</tr>
<tr>
<td>PREVIOUS_FISCAL_QUARTERS_LAST_FISCAL_YEAR</td>
<td>PQLFY</td>
</tr>
<tr>
<td>Value</td>
<td>RelativeDate.dateRangeId Property</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>PREVIOUS_FISCAL_QUARTERS_THIS_FISCAL_YEAR</td>
<td>PQTFY</td>
</tr>
<tr>
<td>PREVIOUS_MONTHS_LAST_FISCAL_HALF</td>
<td>PMLFH</td>
</tr>
<tr>
<td>PREVIOUS_MONTHS_LAST_FISCAL_QUARTER</td>
<td>PMLFQ</td>
</tr>
<tr>
<td>PREVIOUS_MONTHS_LAST_FISCAL_YEAR</td>
<td>PMLFY</td>
</tr>
<tr>
<td>PREVIOUS_MONTHS SAME_FISCAL_HALF LAST_FISCAL_YEAR</td>
<td>PMSFHLFY</td>
</tr>
<tr>
<td>PREVIOUS_MONTHS THIS_FISCAL_HALF</td>
<td>PMTFH</td>
</tr>
<tr>
<td>PREVIOUS_MONTHS THIS_FISCAL_QUARTER</td>
<td>PMTFQ</td>
</tr>
<tr>
<td>PREVIOUS_MONTHS THIS_FISCAL_YEAR</td>
<td>PMTFY</td>
</tr>
<tr>
<td>PREVIOUS_ONE_DAY</td>
<td>OD</td>
</tr>
<tr>
<td>PREVIOUS_ONE_HALF</td>
<td>OH</td>
</tr>
<tr>
<td>PREVIOUS_ONE_MONTH</td>
<td>OM</td>
</tr>
<tr>
<td>PREVIOUS_ONE_QUARTER</td>
<td>OQ</td>
</tr>
<tr>
<td>PREVIOUS_ONE_WEEK</td>
<td>OW</td>
</tr>
<tr>
<td>PREVIOUS_ONE_YEAR</td>
<td>OY</td>
</tr>
<tr>
<td>PREVIOUS_ROLLING_HALF</td>
<td>PRH</td>
</tr>
<tr>
<td>PREVIOUS_ROLLING_QUARTER</td>
<td>PRQ</td>
</tr>
<tr>
<td>PREVIOUS_ROLLING_YEAR</td>
<td>PRY</td>
</tr>
<tr>
<td>SAME_DAY_FISCAL_QUARTER BEFORE_LAST</td>
<td>SDFQBL</td>
</tr>
<tr>
<td>SAME_DAY_FISCAL_YEAR BEFORE_LAST</td>
<td>SDFYBL</td>
</tr>
<tr>
<td>SAME_DAY_LAST_FISCAL_QUARTER</td>
<td>SDLFQ</td>
</tr>
<tr>
<td>SAME_DAY_LAST_FISCAL_YEAR</td>
<td>SDLFY</td>
</tr>
<tr>
<td>SAME_DAY_LAST_MONTH</td>
<td>SDLM</td>
</tr>
<tr>
<td>SAME_DAY_LAST_WEEK</td>
<td>SDLW</td>
</tr>
<tr>
<td>SAME_MONTH_FISCAL_QUARTER BEFORE LAST</td>
<td>SMFQBL</td>
</tr>
<tr>
<td>SAME_FISCAL_HALF LAST FISCAL_YEAR TO DATE</td>
<td>SFHLFY</td>
</tr>
<tr>
<td>SAME_FISCAL_HALF LAST FISCAL_YEAR TO DATE</td>
<td>SFHLFYTD</td>
</tr>
<tr>
<td>SAME_FISCAL_QUARTER FISCAL_YEAR BEFORE LAST</td>
<td>SFQFYBL</td>
</tr>
<tr>
<td>SAME_FISCAL_QUARTER LAST FISCAL_YEAR TO DATE</td>
<td>SFQLFYTD</td>
</tr>
<tr>
<td>SAME_MONTH_FISCAL_QUARTER BEFORE LAST</td>
<td>SMFQBL</td>
</tr>
<tr>
<td>Value</td>
<td>RelativeDate.dateRangeId Property</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>SAME_MONTH_FISCAL_YEAR_BEFORE_LAST</td>
<td>SMFYBL</td>
</tr>
<tr>
<td>SAME_MONTH_LAST_FISCAL_QUARTER</td>
<td>SMLFQ</td>
</tr>
<tr>
<td>SAME_MONTH_LAST_FISCAL_QUARTER_TO_DATE</td>
<td>SMLFQTD</td>
</tr>
<tr>
<td>SAME_MONTH_LAST_FISCAL_YEAR</td>
<td>SMLFY</td>
</tr>
<tr>
<td>SAME_MONTH_LAST_FISCAL_YEAR_TO_DATE</td>
<td>SMLFYTD</td>
</tr>
<tr>
<td>SAME_WEEK_FISCAL_YEAR_BEFORE_LAST</td>
<td>SWFYBL</td>
</tr>
<tr>
<td>SAME_WEEK_LAST_FISCAL_YEAR</td>
<td>SWLFY</td>
</tr>
<tr>
<td>SIXTY_DAYS_AGO</td>
<td>DAGO60</td>
</tr>
<tr>
<td>SIXTY_DAYS_FROM_NOW</td>
<td>DFN60</td>
</tr>
<tr>
<td>TEN_DAYS_AGO</td>
<td>DAGO10</td>
</tr>
<tr>
<td>TEN_DAYS_FROM_NOW</td>
<td>DFN10</td>
</tr>
<tr>
<td>THIRTY_DAYS_AGO</td>
<td>DAGO30</td>
</tr>
<tr>
<td>THIRTY_DAYS_FROM_NOW</td>
<td>DFN30</td>
</tr>
<tr>
<td>THIS_BUSINESS_WEEK</td>
<td>TBW</td>
</tr>
<tr>
<td>THIS_FISCAL_HALF</td>
<td>TFH</td>
</tr>
<tr>
<td>THIS_FISCAL_HALF_TO_DATE</td>
<td>TFHTD</td>
</tr>
<tr>
<td>THIS_FISCAL_QUARTER</td>
<td>TFQ</td>
</tr>
<tr>
<td>THIS_FISCAL_QUARTER_TO_DATE</td>
<td>TFQTD</td>
</tr>
<tr>
<td>THIS_FISCAL_YEAR</td>
<td>TFY</td>
</tr>
<tr>
<td>THIS_FISCAL_YEAR_TO_DATE</td>
<td>TFYTD</td>
</tr>
<tr>
<td>THIS_MONTH</td>
<td>TM</td>
</tr>
<tr>
<td>THIS_MONTH_TO_DATE</td>
<td>TMTD</td>
</tr>
<tr>
<td>THIS_ROLLING_HALF</td>
<td>TRH</td>
</tr>
<tr>
<td>THIS_ROLLING_QUARTER</td>
<td>TRQ</td>
</tr>
<tr>
<td>THIS_ROLLING_YEAR</td>
<td>TRY</td>
</tr>
<tr>
<td>THIS_WEEK</td>
<td>TW</td>
</tr>
<tr>
<td>THIS_WEEK_TO_DATE</td>
<td>TWTD</td>
</tr>
<tr>
<td>THIS_YEAR</td>
<td>TY</td>
</tr>
<tr>
<td>THIS_YEAR_TO_DATE</td>
<td>TYTD</td>
</tr>
<tr>
<td>THREE_DAYS_AGO</td>
<td>DAGO3</td>
</tr>
<tr>
<td>THREE_DAYS_FROM_NOW</td>
<td>DFN3</td>
</tr>
<tr>
<td>THREE_FISCAL_QUARTERS_AGO</td>
<td>FQB</td>
</tr>
</tbody>
</table>
### Value

<table>
<thead>
<tr>
<th>Value</th>
<th>RelativeDate.dateRangeId Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>THREE_FISCAL_QUARTERS_AGO_TO_DATE</td>
<td>FQBTD</td>
</tr>
<tr>
<td>THREE_FISCAL_YEARS_AGO</td>
<td>FYB</td>
</tr>
<tr>
<td>THREE_FISCAL_YEARS_AGO_TO_DATE</td>
<td>FYBTD</td>
</tr>
<tr>
<td>THREE_MONTHS_AGO</td>
<td>MB</td>
</tr>
<tr>
<td>THREE_MONTHS_AGO_TO_DATE</td>
<td>MBTD</td>
</tr>
<tr>
<td>TODAY</td>
<td>TODAY</td>
</tr>
<tr>
<td>TODAY_TO_END_OF_THIS_MONTH</td>
<td>TODAYTTM</td>
</tr>
<tr>
<td>TOMORROW</td>
<td>TOMORROW</td>
</tr>
<tr>
<td>TWO_DAYS_AGO</td>
<td>DAGO2</td>
</tr>
<tr>
<td>TWO_DAYS_FROM_NOW</td>
<td>DFN2</td>
</tr>
<tr>
<td>WEEK_AFTER_NEXT</td>
<td>WAN</td>
</tr>
<tr>
<td>WEEK_AFTER_NEXT_TO_DATE</td>
<td>WANTD</td>
</tr>
<tr>
<td>WEEK_BEFORE_LAST</td>
<td>WBL</td>
</tr>
<tr>
<td>WEEK_BEFORE_LAST_TO_DATE</td>
<td>WBLTD</td>
</tr>
<tr>
<td>YESTERDAY</td>
<td>YESTERDAY</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...

var todayCondition = myQuery.createCondition({
  fieldId: 'trandate',
  operator: query.Operator.BEFORE,
  values: query.RelativeDateRange.TODAY
});

...

// Add additional code
```

### query.ReturnType

**Enum Description**

Holds the string values for the formula return types supported with the N/query Module.

This enum is used to pass the formula return type argument to `Query.createColumn(options)`, `Component.createColumn(options)`, `Query.createCondition(options)`, and `Component.createCondition(options)`.

For more information on formulas, see the help topics SuiteAnalytics Workbook, SQL Expressions, and Search Formula Examples and Tips.
Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module Members</td>
<td>N/query Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

## Values

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANY</td>
</tr>
<tr>
<td>BOOLEAN</td>
</tr>
<tr>
<td>CURRENCY</td>
</tr>
<tr>
<td>DATE</td>
</tr>
<tr>
<td>DATETIME</td>
</tr>
<tr>
<td>DURATION</td>
</tr>
<tr>
<td>FLOAT</td>
</tr>
<tr>
<td>INTEGER</td>
</tr>
<tr>
<td>KEY</td>
</tr>
<tr>
<td>RELATIONSHIP</td>
</tr>
<tr>
<td>STRING</td>
</tr>
<tr>
<td>UNKNOWN</td>
</tr>
</tbody>
</table>

## Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.

```javascript
// Add additional code
...
var myTransactionQuery = query.create({
    type: query.Type.TRANSACTION
});

var myFormulaColumn = myTransactionQuery.createColumn({
    type: query.ReturnType.CURRENCY,
    formula: `(amount) * 125`
});
...
```
query.SortLocale

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for sort locales supported with the N/query Module. This enum is used to pass the locale argument to Query.createSort(options) and Component.createSort(options).</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module Members</td>
<td>N/query Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Values**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ARABIC</td>
</tr>
<tr>
<td>ARABIC_ABJ_MATCH</td>
</tr>
<tr>
<td>ARABIC_ABJ_MATCH_CI</td>
</tr>
<tr>
<td>ARABIC_ABJ_SORT</td>
</tr>
<tr>
<td>ARABIC_ABJ_SORT_CI</td>
</tr>
<tr>
<td>ARABIC_CI</td>
</tr>
<tr>
<td>ARABIC_MATCH</td>
</tr>
<tr>
<td>ARABIC_MATCH_CI</td>
</tr>
<tr>
<td>ASCII7</td>
</tr>
<tr>
<td>ASCII7_CI</td>
</tr>
<tr>
<td>AZERBAIJANI</td>
</tr>
<tr>
<td>AZERBAIJANI_CI</td>
</tr>
<tr>
<td>BENGALI</td>
</tr>
<tr>
<td>BENGALI_CI</td>
</tr>
<tr>
<td>BIG5</td>
</tr>
<tr>
<td>BIG5_CI</td>
</tr>
<tr>
<td>BINARY</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>BINARY_CI</td>
</tr>
<tr>
<td>BULGARIAN</td>
</tr>
<tr>
<td>BULGARIAN_CI</td>
</tr>
<tr>
<td>CANADIAN_M</td>
</tr>
<tr>
<td>CATALAN</td>
</tr>
<tr>
<td>CATALAN_CI</td>
</tr>
<tr>
<td>CROATIAN</td>
</tr>
<tr>
<td>CROATIAN_CI</td>
</tr>
<tr>
<td>CS_CZ</td>
</tr>
<tr>
<td>CZECH</td>
</tr>
<tr>
<td>CZECH_CI</td>
</tr>
<tr>
<td>CZECH_PUNCTUATION</td>
</tr>
<tr>
<td>CZECH_PUNCTUATION_CI</td>
</tr>
<tr>
<td>DANISH</td>
</tr>
<tr>
<td>DANISH_CI</td>
</tr>
<tr>
<td>DANISH_M</td>
</tr>
<tr>
<td>DA_DK</td>
</tr>
<tr>
<td>DE_DE</td>
</tr>
<tr>
<td>DUTCH</td>
</tr>
<tr>
<td>DUTCH_CI</td>
</tr>
<tr>
<td>EBCDIC</td>
</tr>
<tr>
<td>EBCDIC_CI</td>
</tr>
<tr>
<td>EEC_EURO</td>
</tr>
<tr>
<td>EEC_UEUROPA3</td>
</tr>
<tr>
<td>EEC_UEUROPA3_CI</td>
</tr>
<tr>
<td>EEC_EURO_CI</td>
</tr>
<tr>
<td>EN</td>
</tr>
<tr>
<td>EN_AU</td>
</tr>
<tr>
<td>EN_CA</td>
</tr>
<tr>
<td>EN_GB</td>
</tr>
<tr>
<td>EN_US</td>
</tr>
<tr>
<td>ESTONIAN</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>ESTONIAN_CI</td>
</tr>
<tr>
<td>ES_AR</td>
</tr>
<tr>
<td>ES_ES</td>
</tr>
<tr>
<td>FINNISH</td>
</tr>
<tr>
<td>FINNISH_CI</td>
</tr>
<tr>
<td>FRENCH</td>
</tr>
<tr>
<td>FRENCH_AI</td>
</tr>
<tr>
<td>FRENCH_CI</td>
</tr>
<tr>
<td>FRENCH_M</td>
</tr>
<tr>
<td>FR_CA</td>
</tr>
<tr>
<td>FR_FR</td>
</tr>
<tr>
<td>GBK</td>
</tr>
<tr>
<td>GBK_AI</td>
</tr>
<tr>
<td>GBK_CI</td>
</tr>
<tr>
<td>GENERIC_M</td>
</tr>
<tr>
<td>GERMAN</td>
</tr>
<tr>
<td>GERMAN_AI</td>
</tr>
<tr>
<td>GERMAN_CI</td>
</tr>
<tr>
<td>GERMAN_DIN</td>
</tr>
<tr>
<td>GERMAN_DIN_AI</td>
</tr>
<tr>
<td>GERMAN_DIN_CI</td>
</tr>
<tr>
<td>GREEK</td>
</tr>
<tr>
<td>GREEK_AI</td>
</tr>
<tr>
<td>GREEK_CI</td>
</tr>
<tr>
<td>HEBREW</td>
</tr>
<tr>
<td>HEBREW_AI</td>
</tr>
<tr>
<td>HEBREW_CI</td>
</tr>
<tr>
<td>HE_IL</td>
</tr>
<tr>
<td>HKSCS</td>
</tr>
<tr>
<td>HKSCS_AI</td>
</tr>
<tr>
<td>HKSCS_CI</td>
</tr>
<tr>
<td>HUNGARIAN</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>HUNGARIAN_AI</td>
</tr>
<tr>
<td>HUNGARIAN_CI</td>
</tr>
<tr>
<td>ICELANDIC</td>
</tr>
<tr>
<td>ICELANDIC_AI</td>
</tr>
<tr>
<td>ICELANDIC_CI</td>
</tr>
<tr>
<td>INDONESIAN</td>
</tr>
<tr>
<td>INDONESIAN_AI</td>
</tr>
<tr>
<td>INDONESIAN_CI</td>
</tr>
<tr>
<td>ITALIAN</td>
</tr>
<tr>
<td>ITALIAN_AI</td>
</tr>
<tr>
<td>ITALIAN_CI</td>
</tr>
<tr>
<td>IT_IT</td>
</tr>
<tr>
<td>JAPANESE_M</td>
</tr>
<tr>
<td>JA_JP</td>
</tr>
<tr>
<td>KOREAN_M</td>
</tr>
<tr>
<td>KO_KR</td>
</tr>
<tr>
<td>LATIN</td>
</tr>
<tr>
<td>LATIN_AI</td>
</tr>
<tr>
<td>LATIN_CI</td>
</tr>
<tr>
<td>LATVIAN</td>
</tr>
<tr>
<td>LATVIAN_AI</td>
</tr>
<tr>
<td>LATVIAN_CI</td>
</tr>
<tr>
<td>LITHUANIAN</td>
</tr>
<tr>
<td>LITHUANIAN_AI</td>
</tr>
<tr>
<td>LITHUANIAN_CI</td>
</tr>
<tr>
<td>MALAY</td>
</tr>
<tr>
<td>MALAY_AI</td>
</tr>
<tr>
<td>MALAY_CI</td>
</tr>
<tr>
<td>NL_NL</td>
</tr>
<tr>
<td>NORWEGIAN</td>
</tr>
<tr>
<td>NORWEGIAN_AI</td>
</tr>
<tr>
<td>NORWEGIAN_CI</td>
</tr>
<tr>
<td>Value</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>POLISH</td>
</tr>
<tr>
<td>POLISH.AI</td>
</tr>
<tr>
<td>POLISH_CI</td>
</tr>
<tr>
<td>PT_BR</td>
</tr>
<tr>
<td>PUNCTUATION</td>
</tr>
<tr>
<td>PUNCTUATION.AI</td>
</tr>
<tr>
<td>PUNCTUATION_CI</td>
</tr>
<tr>
<td>ROMANIAN</td>
</tr>
<tr>
<td>ROMANIAN.AI</td>
</tr>
<tr>
<td>ROMANIAN_CI</td>
</tr>
<tr>
<td>RUSSIAN</td>
</tr>
<tr>
<td>RUSSIAN.AI</td>
</tr>
<tr>
<td>RUSSIAN_CI</td>
</tr>
<tr>
<td>RU_RU</td>
</tr>
<tr>
<td>SCHINESE_PINYIN_M</td>
</tr>
<tr>
<td>SCHINESE_RADICAL_M</td>
</tr>
<tr>
<td>SCHINESE_STROKE_M</td>
</tr>
<tr>
<td>SLOVAK</td>
</tr>
<tr>
<td>SLOVAK.AI</td>
</tr>
<tr>
<td>SLOVAK_CI</td>
</tr>
<tr>
<td>SLOVENIAN</td>
</tr>
<tr>
<td>SLOVENIAN.AI</td>
</tr>
<tr>
<td>SLOVENIAN_CI</td>
</tr>
<tr>
<td>SPANISH</td>
</tr>
<tr>
<td>SPANISH.AI</td>
</tr>
<tr>
<td>SPANISH_CI</td>
</tr>
<tr>
<td>SPANISH_M</td>
</tr>
<tr>
<td>SV_SE</td>
</tr>
<tr>
<td>SWEDISH</td>
</tr>
<tr>
<td>SWEDISH.AI</td>
</tr>
<tr>
<td>SWEDISH_CI</td>
</tr>
<tr>
<td>SWISS</td>
</tr>
</tbody>
</table>
### Value

- SWISS_AI
- SWISS_CI
- TCHINESE_RADICAL_M
- TCHINESE_STROKE_M
- THAI_M
- TH_TH
- TR_TR
- TURKISH
- TURKISH_AI
- TURKISH_CI
- UKRAINIAN
- UKRAINIAN_AI
- UKRAINIAN_CI
- UNICODE_BINARY
- UNICODE_BINARY_AI
- UNICODE_BINARY_CI
- VIETNAMESE
- VIETNAMESE_AI
- VIETNAMESE_CI
- WEST_EUROPEAN
- WEST_EUROPEAN_AI
- WEST_EUROPEAN_CI
- ZH_CN
- ZH_TW

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/query Module Script Samples](#).

```javascript
// Add additional code
...
var myCustomerQuery = query.create({
  type: query.Type.CUSTOMER
});

myCustomerQuery.columns = [
```
myCustomerQuery.createColumn({
    fieldId: 'entityid'
})
]

myCustomerQuery.sort = [
    myCustomerQuery.createSort({
        column: myCustomerQuery.columns[0],
        locale: query.SortLocale.EN_CA
    })
];
...
// Add additional code

## query.Type

### Important: The N/query module supports the same record types that are supported by the SuiteAnalytics Workbook interface. For more information, see the help topic Available Records.

| Enum Description | Holds the string values for search types used in the query definition. This enum is used to pass the initial search type argument to `query.create(options)`.
|------------------|--------------------------------------------------------------------------------------------------|

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Type</th>
<th>enum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/query Module</td>
</tr>
<tr>
<td>Sibling Module</td>
<td>N/query Module Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.1</td>
</tr>
</tbody>
</table>

## Values

**Note:** Before using these values, consider the following:

- A search type is not the same as a record type. The supported search types listed below do not necessarily correspond with the supported record types listed in the N/record Module.
- Depending on your account and role, some of these values might not be available.

<table>
<thead>
<tr>
<th>Enum Value</th>
<th>Sets Query.type Property To</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNT</td>
<td>account</td>
</tr>
<tr>
<td>ACCOUNTING_CONTEXT</td>
<td>accountingcontext</td>
</tr>
<tr>
<td>ACCOUNTING_PERIOD</td>
<td>accountingperiod</td>
</tr>
<tr>
<td>ADVANCED_REV_REC_PLUGIN</td>
<td>advancedrevrecplugin</td>
</tr>
<tr>
<td>ADV_INTERCOMPANY_JOURNAL_ENTRY</td>
<td>advintercompanyjournalentry</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>ALLOCATION_METHOD</td>
<td>allocationmethod</td>
</tr>
<tr>
<td>AMORTIZATION_SCHEDULE</td>
<td>amortizationschedule</td>
</tr>
<tr>
<td>AMORTIZATION_TEMPLATE</td>
<td>amortizationtemplate</td>
</tr>
<tr>
<td>ANOTHER_HIERARCHY_RECORD</td>
<td>anotherhierarchyrecord</td>
</tr>
<tr>
<td>BANK_CONNECTIVITY_PLUGIN</td>
<td>bankconnectivityplugin</td>
</tr>
<tr>
<td>BILLING_CLASS</td>
<td>billingclass</td>
</tr>
<tr>
<td>BILLING_SCHEDULE</td>
<td>billschedule</td>
</tr>
<tr>
<td>BRANCHRECORD</td>
<td>branchrecord</td>
</tr>
<tr>
<td>BUDGETCATEGORY</td>
<td>budgetcategory</td>
</tr>
<tr>
<td>BUDGETEXCHANGERATE</td>
<td>budgetexchangerate</td>
</tr>
<tr>
<td>BUDGETIMPORT</td>
<td>budgetimport</td>
</tr>
<tr>
<td>BUDGETS</td>
<td>budgets</td>
</tr>
<tr>
<td>BULK_PROC_SUBMISSION</td>
<td>bulkprocsubmission</td>
</tr>
<tr>
<td>BUNDLE_INSTALLATION_SCRIPT</td>
<td>bundleinstallationscript</td>
</tr>
<tr>
<td>BUNDLE_INSTALLATION_SCRIPT_DEPLOYMENT</td>
<td>bundleinstallationscriptdeployment</td>
</tr>
<tr>
<td>BUYING_REASON</td>
<td>buyingreason</td>
</tr>
<tr>
<td>BUYING_TIME_FRAME</td>
<td>buyingtimeframe</td>
</tr>
<tr>
<td>CALENDAR_EVENT</td>
<td>calendarevent</td>
</tr>
<tr>
<td>CAMPAIGN_AUDIENCE</td>
<td>campaignaudience</td>
</tr>
<tr>
<td>CAMPAIGN_CATEGORY</td>
<td>campaigncategory</td>
</tr>
<tr>
<td>CAMPAIGN_CHANNEL</td>
<td>campaignchannel</td>
</tr>
<tr>
<td>CAMPAIGN_EMAIL_ADDRESS</td>
<td>campaignemailaddress</td>
</tr>
<tr>
<td>CAMPAIGN_EVENT</td>
<td>campaignevent</td>
</tr>
<tr>
<td>CAMPAIGN_FAMILY</td>
<td>campaignfamily</td>
</tr>
<tr>
<td>CAMPAIGN_OFFER</td>
<td>campaignoffer</td>
</tr>
<tr>
<td>CAMPAIGN_RESPONSE</td>
<td>campaignresponse</td>
</tr>
<tr>
<td>CAMPAIGN_SEARCH_ENGINE</td>
<td>campaignsearchengine</td>
</tr>
<tr>
<td>CAMPAIGN_TEMPLATE</td>
<td>campaigntemplate</td>
</tr>
<tr>
<td>CAMPAIGN_VERTICAL</td>
<td>campaignvertical</td>
</tr>
<tr>
<td>CASE_PROFILE</td>
<td>caseprofile</td>
</tr>
<tr>
<td>CASH_REFUND</td>
<td>cashrefund</td>
</tr>
<tr>
<td>CASH_SALE</td>
<td>cashsale</td>
</tr>
<tr>
<td>CATEGORY1099MISC</td>
<td>category1099misc</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>CHECK</td>
<td>check</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>classification</td>
</tr>
<tr>
<td>CLIENT_SCRIPT</td>
<td>clientscript</td>
</tr>
<tr>
<td>CLIENT_SCRIPT_DEPLOYMENT</td>
<td>clientscriptdeployment</td>
</tr>
<tr>
<td>CLOB_RECORD</td>
<td>clobrecord</td>
</tr>
<tr>
<td>COMPANY</td>
<td>company</td>
</tr>
<tr>
<td>COMPETITOR</td>
<td>competitor</td>
</tr>
<tr>
<td>COMPOSITE_KEY_SOURCE_RECORD</td>
<td>compositekeysourcerecord</td>
</tr>
<tr>
<td>COMPOSITE_RECORD</td>
<td>compositerecord</td>
</tr>
<tr>
<td>CONSOLIDATEDEXCHANGERATE</td>
<td>consolidatedexchangerate</td>
</tr>
<tr>
<td>CONSOLIDATEDEXCHANGERATEINTERNAL</td>
<td>consolidatedexchangerateinternal</td>
</tr>
<tr>
<td>CONSOLIDATED_RATE_ADJUSTER_PLUGIN</td>
<td>consolidatedrateadjustorplugin</td>
</tr>
<tr>
<td>CONSOLIDATION_ACCOUNT</td>
<td>consolidationaccount</td>
</tr>
<tr>
<td>CONSOLIDATION_ACCOUNT_TYPE</td>
<td>consolidationaccounttype</td>
</tr>
<tr>
<td>CONSOLIDATION_BUDGET_RATE</td>
<td>consolidationbudgetrate</td>
</tr>
<tr>
<td>CONSOLIDATION_CURRENCY</td>
<td>consolidationcurrency</td>
</tr>
<tr>
<td>CONSOLIDATION_RATE</td>
<td>consolidationrate</td>
</tr>
<tr>
<td>CONSOLIDATION_SUBSIDIARY</td>
<td>consolidationsubsidiary</td>
</tr>
<tr>
<td>CONSOLIDATION_TRANSACTION</td>
<td>consolidationtransaction</td>
</tr>
<tr>
<td>CONSUMER_SPECIFIC_RECORD_TYPE</td>
<td>consumerspecificrecordtype</td>
</tr>
<tr>
<td>CONTACT</td>
<td>contact</td>
</tr>
<tr>
<td>CONTACT_CATEGORY</td>
<td>contactcategory</td>
</tr>
<tr>
<td>CONTACT_ROLE</td>
<td>contactrole</td>
</tr>
<tr>
<td>COUPON_CODE</td>
<td>couponcode</td>
</tr>
<tr>
<td>COURSE_RECORD</td>
<td>courserecord</td>
</tr>
<tr>
<td>CREDIT_CARDS</td>
<td>creditcards</td>
</tr>
<tr>
<td>CREDIT_CARD_CHARGE</td>
<td>creditcardcharge</td>
</tr>
<tr>
<td>CREDIT_CARD_REFUND</td>
<td>creditcardrefund</td>
</tr>
<tr>
<td>CREDIT_MEMO</td>
<td>creditmemo</td>
</tr>
<tr>
<td>CRM_TEMPLATE</td>
<td>crmtemplate</td>
</tr>
<tr>
<td>CRM_TEMPLATE_CATEGORY</td>
<td>crmtemplatecategory</td>
</tr>
<tr>
<td>CURRENCY</td>
<td>currency</td>
</tr>
<tr>
<td>CURRENCY_FIELD_RECORD</td>
<td>currencyfieldrecord</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>CURRENCY_FIELD_TYPE</td>
<td>currencyfieldtype</td>
</tr>
<tr>
<td>CURRENCY_RATE</td>
<td>currencyrate</td>
</tr>
<tr>
<td>CUSTOM</td>
<td>custom</td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>customer</td>
</tr>
<tr>
<td>CUSTOMER_CATEGORY</td>
<td>customercategory</td>
</tr>
<tr>
<td>CUSTOMER_CHARGE</td>
<td>customercharge</td>
</tr>
<tr>
<td>CUSTOMER_DEPOSIT</td>
<td>customerdeposit</td>
</tr>
<tr>
<td>CUSTOMER_MESSAGE</td>
<td>customermessage</td>
</tr>
<tr>
<td>CUSTOMER_PAYMENT</td>
<td>customerpayment</td>
</tr>
<tr>
<td>CUSTOMER_REFUND</td>
<td>customerrefund</td>
</tr>
<tr>
<td>CUSTOMER_STATUS</td>
<td>customerstatus</td>
</tr>
<tr>
<td>CUSTOMRECORD1</td>
<td>customrecord1</td>
</tr>
<tr>
<td>CUSTOM_GL_PLUGIN</td>
<td>customgplugin</td>
</tr>
<tr>
<td>CUSTOM_LIST</td>
<td>customlist</td>
</tr>
<tr>
<td>CUSTOM_RECORD_TYPE</td>
<td>customrecordtype</td>
</tr>
<tr>
<td>DATE_FIELD_TYPE</td>
<td>datetype</td>
</tr>
<tr>
<td>DATE_RECORD</td>
<td>daterecord</td>
</tr>
<tr>
<td>DATE_TIME_RECORD</td>
<td>datetimerecord</td>
</tr>
<tr>
<td>DATE_TIME_ZONE</td>
<td>datetimestamp</td>
</tr>
<tr>
<td>DEFAULTING_PORTED_RECORD</td>
<td>defaultingportedrecord</td>
</tr>
<tr>
<td>DEF_VIEW_TEST_RECORD</td>
<td>defviewtestrecord</td>
</tr>
<tr>
<td>DELETED_RECORD</td>
<td>deletedrecord</td>
</tr>
<tr>
<td>DEPARTMENT</td>
<td>department</td>
</tr>
<tr>
<td>DEPOSIT</td>
<td>deposit</td>
</tr>
<tr>
<td>DEPOSIT_APPLICATION</td>
<td>depositapplication</td>
</tr>
<tr>
<td>DESCRIPTION_ITEM</td>
<td>descriptionitem</td>
</tr>
<tr>
<td>DEVICE_ID</td>
<td>deviceid</td>
</tr>
<tr>
<td>DISABLEDCALLHANNELFORMTESTRECORD</td>
<td>disabledchannelformtestrecord</td>
</tr>
<tr>
<td>DISCOUNT_ITEM</td>
<td>discountitem</td>
</tr>
<tr>
<td>DISPLAY_INACTIVE_TEST_RECORD</td>
<td>displayinactiveetestrecord</td>
</tr>
<tr>
<td>DOMAIN</td>
<td>domain</td>
</tr>
<tr>
<td>DOWNLOAD_ITEM</td>
<td>downloaditem</td>
</tr>
<tr>
<td>DURATION_RECORD</td>
<td>durationrecord</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>EMAIL_CAPTURE_PLUGIN</td>
<td>emailcaptureplugin</td>
</tr>
<tr>
<td>EMAIL_TEMPLATE</td>
<td>emailetmaeltemplate</td>
</tr>
<tr>
<td>EMPLOYEE</td>
<td>employee</td>
</tr>
<tr>
<td>EMPLOYEE_LIST</td>
<td>employeeList</td>
</tr>
<tr>
<td>EMPLOYEE_STATUS</td>
<td>employeestatus</td>
</tr>
<tr>
<td>END_TO_END_TIME</td>
<td>endtoendtime</td>
</tr>
<tr>
<td>ENTITY</td>
<td>entity</td>
</tr>
<tr>
<td>ENTITY_GROUP</td>
<td>entitygroup</td>
</tr>
<tr>
<td>ESCALATION_TERRITORY</td>
<td>escalationterritory</td>
</tr>
<tr>
<td>ESTIMATE</td>
<td>estimate</td>
</tr>
<tr>
<td>EXAMPLE_TRANSACTION</td>
<td>exampletransaction</td>
</tr>
<tr>
<td>EXPENSECATEGORY</td>
<td>expensescategory</td>
</tr>
<tr>
<td>EXPENSE_REPORT</td>
<td>expensereport</td>
</tr>
<tr>
<td>EXPOSURENOTLIMITEDRECORD</td>
<td>exposurenotlimitedrecord</td>
</tr>
<tr>
<td>FACULTYRECORD</td>
<td>facultyrecord</td>
</tr>
<tr>
<td>FAX TEMPLATE</td>
<td>faxtemplate</td>
</tr>
<tr>
<td>FIELD_LABEL</td>
<td>fieldlabel</td>
</tr>
<tr>
<td>FILE</td>
<td>file</td>
</tr>
<tr>
<td>FLOAT_NUMBERS_TEST_RECORD</td>
<td>floatnumberstestrecord</td>
</tr>
<tr>
<td>FORECAST</td>
<td>forecast</td>
</tr>
<tr>
<td>FORMULA_POLYMORPHIC_RECORD</td>
<td>formulapolymorphicrecord</td>
</tr>
<tr>
<td>FORMULA_RECORD</td>
<td>formularecord</td>
</tr>
<tr>
<td>FULFILLMENT_EXCEPTION_REASON</td>
<td>fulfillmentexceptionreason</td>
</tr>
<tr>
<td>FX_REVAL</td>
<td>fxreval</td>
</tr>
<tr>
<td>GATEWAY_NOTIFICATION</td>
<td>gatewaynotification</td>
</tr>
<tr>
<td>GENERAL_ALLOCATION_SCHEDULE</td>
<td>generalallocationschedule</td>
</tr>
<tr>
<td>GENERIC_RESOURCE</td>
<td>genericresource</td>
</tr>
<tr>
<td>GENERIC_TEST_RECORD</td>
<td>genericitestrecord</td>
</tr>
<tr>
<td>GIFT_CERTIFICATE</td>
<td>giftcertificate</td>
</tr>
<tr>
<td>GIFT_CERTIFICATE_ITEM</td>
<td>giftcertificateitem</td>
</tr>
<tr>
<td>HIERARCHY_RECORD</td>
<td>hierarchyrecord</td>
</tr>
<tr>
<td>HYBRID_RECORD_LOG</td>
<td>hybridrecordlog</td>
</tr>
<tr>
<td>INCOTERM</td>
<td>incoterm</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>INTEGRATION_APP</td>
<td>integrationapp</td>
</tr>
<tr>
<td>INTERNAL_ID_TEST_RECORD</td>
<td>internalidtestrecord</td>
</tr>
<tr>
<td>INVENTORY_ADJUSTMENT</td>
<td>inventoryadjustment</td>
</tr>
<tr>
<td>INVENTORY_DISTRIBUTION</td>
<td>inventorydistribution</td>
</tr>
<tr>
<td>INVENTORY_ITEM</td>
<td>inventoryitem</td>
</tr>
<tr>
<td>INVENTORY_TRANSFER</td>
<td>inventorytransfer</td>
</tr>
<tr>
<td>INVENTORY_WORKSHEET</td>
<td>inventoryworksheet</td>
</tr>
<tr>
<td>INVOICE</td>
<td>invoice</td>
</tr>
<tr>
<td>INVT_ITEM_PRICE_HISTORY</td>
<td>invtitempricehistory</td>
</tr>
<tr>
<td>ISSUE</td>
<td>issue</td>
</tr>
<tr>
<td>ISSUE_EXTERNAL_STATUS</td>
<td>issueexternalstatus</td>
</tr>
<tr>
<td>ISSUE_PRIORITY</td>
<td>issuepriority</td>
</tr>
<tr>
<td>ISSUE_PRODUCT</td>
<td>issueproduct</td>
</tr>
<tr>
<td>ISSUE_REPRODUCIBILITY</td>
<td>issuereproducibility</td>
</tr>
<tr>
<td>ISSUE_ROLE</td>
<td>issuerole</td>
</tr>
<tr>
<td>ISSUE_SEVERITY</td>
<td>issueseverity</td>
</tr>
<tr>
<td>ISSUE_SOURCE</td>
<td>issuesource</td>
</tr>
<tr>
<td>ISSUE_STATUS</td>
<td>issuestatus</td>
</tr>
<tr>
<td>ISSUE_TAG</td>
<td>issuetag</td>
</tr>
<tr>
<td>ISSUE_TRACK_CODE</td>
<td>issuetrackcode</td>
</tr>
<tr>
<td>ISSUE_TYPE</td>
<td>issuetype</td>
</tr>
<tr>
<td>ITEM</td>
<td>item</td>
</tr>
<tr>
<td>ITEM_FULFILLMENT</td>
<td>itemfulfillment</td>
</tr>
<tr>
<td>ITEM_GROUP</td>
<td>itemgroup</td>
</tr>
<tr>
<td>ITEM_RECEIPT</td>
<td>itemreceipt</td>
</tr>
<tr>
<td>I_P_RESTRICTIONS</td>
<td>iprestrictions</td>
</tr>
<tr>
<td>JOB</td>
<td>job</td>
</tr>
<tr>
<td>JOB_RESOURCE_ROLE</td>
<td>jobresourcerole</td>
</tr>
<tr>
<td>JOB_STATUS</td>
<td>jobstatus</td>
</tr>
<tr>
<td>JOB_TYPE</td>
<td>jobtype</td>
</tr>
<tr>
<td>JOURNAL</td>
<td>journal</td>
</tr>
<tr>
<td>KIT_ITEM</td>
<td>kititem</td>
</tr>
<tr>
<td>KNOWLEDGE_BASE</td>
<td>knowledgebase</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>LOCATION</td>
<td>location</td>
</tr>
<tr>
<td>LOCATION_COSTING_GROUP</td>
<td>locationcostinggroup</td>
</tr>
<tr>
<td>LOGIN_AUDIT</td>
<td>loginaudit</td>
</tr>
<tr>
<td>MAIL_TEMPLATE</td>
<td>mailtemplate</td>
</tr>
<tr>
<td>MAP_REDUCE_SCRIPT</td>
<td>mapreducescript</td>
</tr>
<tr>
<td>MAP_REDUCE_SCRIPT_DEPLOYMENT</td>
<td>mapreducescriptdeployment</td>
</tr>
<tr>
<td>MARKUP_ITEM</td>
<td>markupitem</td>
</tr>
<tr>
<td>MASS_UPDATE_SCRIPT</td>
<td>massupdatescript</td>
</tr>
<tr>
<td>MASS_UPDATE_SCRIPT_DEPLOYMENT</td>
<td>massupdatescriptdeployment</td>
</tr>
<tr>
<td>MATERIALIZED_HIERARCHY_RECORD</td>
<td>materializedhierarchyrrecord</td>
</tr>
<tr>
<td>MEDIA_ITEM_FOLDER</td>
<td>mediaitemfolder</td>
</tr>
<tr>
<td>MEM_DOC</td>
<td>memdoc</td>
</tr>
<tr>
<td>MEM_DOC_TRANSACTION_TEMPLATE</td>
<td>memdoctransactiontemplate</td>
</tr>
<tr>
<td>MESSAGE</td>
<td>message</td>
</tr>
<tr>
<td>NAMED_GROUP_RECORD</td>
<td>namedgrouprecord</td>
</tr>
<tr>
<td>NEXUS</td>
<td>nexus</td>
</tr>
<tr>
<td>NON_INVENTORY_PURCHASE_ITEM</td>
<td>noninventorypurchaseitem</td>
</tr>
<tr>
<td>NON_INVENTORY_RESALE_ITEM</td>
<td>noninventoryresaleitem</td>
</tr>
<tr>
<td>NON_INVENTORY_SALE_ITEM</td>
<td>noninventorysaleitem</td>
</tr>
<tr>
<td>NOTE</td>
<td>note</td>
</tr>
<tr>
<td>NOTE_TYPE</td>
<td>notetype</td>
</tr>
<tr>
<td>NUMERIC_RECORD</td>
<td>numericrecord</td>
</tr>
<tr>
<td>ONLINE_CASE_FORM</td>
<td>onlinecaseform</td>
</tr>
<tr>
<td>ONLINE_FORM_TEMPLATE</td>
<td>onlineformtemplate</td>
</tr>
<tr>
<td>ONLINE_LEAD_FORM</td>
<td>onlineleadform</td>
</tr>
<tr>
<td>OPPORTUNITY</td>
<td>opportunity</td>
</tr>
<tr>
<td>OTHER_CHARGE_PURCHASE_ITEM</td>
<td>otherchargepurchaseitem</td>
</tr>
<tr>
<td>OTHER_CHARGE_RESALE_ITEM</td>
<td>otherchargeresaleitem</td>
</tr>
<tr>
<td>OTHER_CHARGE_SALE_ITEM</td>
<td>otherchargesaleitem</td>
</tr>
<tr>
<td>OTHER_NAME</td>
<td>othername</td>
</tr>
<tr>
<td>OTHER_NAME_CATEGORY</td>
<td>othernamecategory</td>
</tr>
<tr>
<td>PAGE</td>
<td>page</td>
</tr>
<tr>
<td>PAGINATION_RECORD</td>
<td>paginationrecord</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>PARTNER</td>
<td>partner</td>
</tr>
<tr>
<td>PARTNER_CATEGORY</td>
<td>partnercategory</td>
</tr>
<tr>
<td>PAYCHECK</td>
<td>paycheck</td>
</tr>
<tr>
<td>PAYMENT_EVENT</td>
<td>paymentevent</td>
</tr>
<tr>
<td>PAYMENT_GATEWAY_PLUGIN</td>
<td>paymentgatewayplugin</td>
</tr>
<tr>
<td>PAYMENT_ITEM</td>
<td>paymentitem</td>
</tr>
<tr>
<td>PAYMENT_METHOD</td>
<td>paymentmethod</td>
</tr>
<tr>
<td>PAYMENT_PROCESSING_PROFILE</td>
<td>paymentprocessingprofile</td>
</tr>
<tr>
<td>PAYROLL_ITEM</td>
<td>payrollitem</td>
</tr>
<tr>
<td>PDF_TEMPLATE</td>
<td>pdftemplate</td>
</tr>
<tr>
<td>PERSISTED_RECORD</td>
<td>persistedrecord</td>
</tr>
<tr>
<td>PERSISTED_RECORD_FULL_JOIN</td>
<td>persistedrecordfulljoin</td>
</tr>
<tr>
<td>PERSISTED_RECORD_INVALID_TABLE</td>
<td>persistedrecordinvalidtable</td>
</tr>
<tr>
<td>PERSISTED_RECORD_NO_CREATE</td>
<td>persistedrecordnocreate</td>
</tr>
<tr>
<td>PERSISTED_RECORD_NO_DELETE</td>
<td>persistedrecordnodelete</td>
</tr>
<tr>
<td>PERSISTED_RECORD_NO_EDIT</td>
<td>persistedrecordnoedit</td>
</tr>
<tr>
<td>PERSISTED_RECORD_NO_LOAD</td>
<td>persistedrecordnoload</td>
</tr>
<tr>
<td>PERSISTED_RECORD_RIGHT_JOIN</td>
<td>persistedrecordrightjoin</td>
</tr>
<tr>
<td>PERSISTED_RECORD_SIMPLE_OPTIONS</td>
<td>persistedrecordsimpleoptions</td>
</tr>
<tr>
<td>PERSISTED_RECORD_U_Q_KEY_REF</td>
<td>persistedrecorduqkeyref</td>
</tr>
<tr>
<td>PERSISTED_RECORD_U_Q_KEY_REF_TYPE</td>
<td>persistedrecorduqkeyreftype</td>
</tr>
<tr>
<td>PHONE_CALL</td>
<td>phonecall</td>
</tr>
<tr>
<td>PLUG_IN_TYPE</td>
<td>plugintype</td>
</tr>
<tr>
<td>PLUG_IN_TYPE_IMPL</td>
<td>plugintypeimpl</td>
</tr>
<tr>
<td>PORTLET</td>
<td>portlet</td>
</tr>
<tr>
<td>PORTLET_DEPLOYMENT</td>
<td>portletdeployment</td>
</tr>
<tr>
<td>PRICE_LEVEL</td>
<td>pricelevel</td>
</tr>
<tr>
<td>PRICING</td>
<td>pricing</td>
</tr>
<tr>
<td>PRICING_GROUP</td>
<td>pricinggroup</td>
</tr>
<tr>
<td>PRIMARY_RECORD</td>
<td>primaryrecord</td>
</tr>
<tr>
<td>PROJECT_TASK</td>
<td>projecttask</td>
</tr>
<tr>
<td>PROJECT_TEMPLATE</td>
<td>projecttemplate</td>
</tr>
<tr>
<td>PROMOTIONS_PLUGIN</td>
<td>promotionsplugin</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>PROMOTION_CODE</td>
<td>promotioncode</td>
</tr>
<tr>
<td>PUBLISHED_SAVED_SEARCH</td>
<td>publishedsavedsearch</td>
</tr>
<tr>
<td>PURCHASE_ORDER</td>
<td>purchaseorder</td>
</tr>
<tr>
<td>PURCHASE_REQUISITION</td>
<td>purchaserequisition</td>
</tr>
<tr>
<td>QUANTITY_PRICING_SCHEDULE</td>
<td>quantitypricingschedule</td>
</tr>
<tr>
<td>QUOTA</td>
<td>quota</td>
</tr>
<tr>
<td>RECENT_RECORD</td>
<td>recentrecord</td>
</tr>
<tr>
<td>RECORD_SERVICE_TEST_RECORD</td>
<td>recordservicetestrecord</td>
</tr>
<tr>
<td>RECORD_TYPE</td>
<td>recordtype</td>
</tr>
<tr>
<td>RECORD_WITH_HIERARCHY_RELATIONSHIP</td>
<td>recordwithhierarchyrelationship</td>
</tr>
<tr>
<td>REDIRECT</td>
<td>redirect</td>
</tr>
<tr>
<td>REGION</td>
<td>region</td>
</tr>
<tr>
<td>RELATIONSHIP_DISPLAY_INACTIVE</td>
<td>relationshipdisplayinactive</td>
</tr>
<tr>
<td>RELATIONSHIP_SELECT_EMPLOYEE_RECORD</td>
<td>relationshipselectemployeerecord</td>
</tr>
<tr>
<td>REPORT_DEFINITION</td>
<td>reportdefinition</td>
</tr>
<tr>
<td>REQUEST_LEVEL_RECORD1</td>
<td>requestlevelrecord1</td>
</tr>
<tr>
<td>REQUEST_LEVEL_RECORD2</td>
<td>requestlevelrecord2</td>
</tr>
<tr>
<td>RESOURCE</td>
<td>resource</td>
</tr>
<tr>
<td>RESTLET</td>
<td>restlet</td>
</tr>
<tr>
<td>RESTLET_DEPLOYMENT</td>
<td>restletdeployment</td>
</tr>
<tr>
<td>RESTRICTIONS_ONCE_REMOVED</td>
<td>restrictionsonceremoved</td>
</tr>
<tr>
<td>RESTRICTIONS_TWICE_REMOVED</td>
<td>restrictionstwiceremoved</td>
</tr>
<tr>
<td>RESTRICTION_ANNOTATION_TEST_RECORD</td>
<td>restrictionannotationtestrecord</td>
</tr>
<tr>
<td>RESTRICTION_TEST_RECORD</td>
<td>restrictiontestrecord</td>
</tr>
<tr>
<td>RETURN_AUTHORIZATION</td>
<td>returnauthorization</td>
</tr>
<tr>
<td>REV_REC_SCHEDULE</td>
<td>revrecschedule</td>
</tr>
<tr>
<td>REV_REC_TEMPLATE</td>
<td>revrectemplate</td>
</tr>
<tr>
<td>ROLE</td>
<td>role</td>
</tr>
<tr>
<td>RSTR_ALT_LOCATION</td>
<td>rstraltlocation</td>
</tr>
<tr>
<td>RSTR_LOCATION</td>
<td>rstrlocation</td>
</tr>
<tr>
<td>RSTR_RECORD</td>
<td>rstrrecord</td>
</tr>
<tr>
<td>SALES</td>
<td>sales</td>
</tr>
<tr>
<td>SALES_ORDER</td>
<td>salesorder</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>SALES_READINESS</td>
<td>salesreadiness</td>
</tr>
<tr>
<td>SALES_ROLE</td>
<td>salesrole</td>
</tr>
<tr>
<td>SALES_TAX_ITEM</td>
<td>saletaxitem</td>
</tr>
<tr>
<td>SALES_TERRITORY</td>
<td>salesterritory</td>
</tr>
<tr>
<td>SALES_TRANSACTION</td>
<td>saletransaction</td>
</tr>
<tr>
<td>SAMPLE_RECORD</td>
<td>samplerecord</td>
</tr>
<tr>
<td>SCHEDULED_SCRIPT</td>
<td>scheduledscript</td>
</tr>
<tr>
<td>SCHEDULED_SCRIPT_DEPLOYMENT</td>
<td>scheduledscriptdeployment</td>
</tr>
<tr>
<td>SCHEDULED_SCRIPT_INSTANCE</td>
<td>scheduledscriptinstance</td>
</tr>
<tr>
<td>SCRIPT</td>
<td>script</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD</td>
<td>scriptingtestrecord</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD2_TARGET</td>
<td>scriptingtestrecordsubrecord2target</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD2_TARGET2</td>
<td>scriptingtestrecordsubrecord2target2</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD3_TARGET</td>
<td>scriptingtestrecordsubrecord3target</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD3_TARGET2</td>
<td>scriptingtestrecordsubrecord3target2</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD4_TARGET</td>
<td>scriptingtestrecordsubrecord4target</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD4_TARGET2</td>
<td>scriptingtestrecordsubrecord4target2</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD_TARGET</td>
<td>scriptingtestrecordsubrecordtarget</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_SUBRECORD_TARGET2</td>
<td>scriptingtestrecordsubrecordtarget2</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_TARGET</td>
<td>scriptingtestrecordtarget</td>
</tr>
<tr>
<td>SCRIPTING_TEST_RECORD_TARGET2</td>
<td>scriptingtestrecordtarget2</td>
</tr>
<tr>
<td>SCRIPT_DEPLOYMENT</td>
<td>scriptdeployment</td>
</tr>
<tr>
<td>SCRIPT_NOTE</td>
<td>scriptnote</td>
</tr>
<tr>
<td>SCRIPT_RECORD_TYPE</td>
<td>scriptrecordtype</td>
</tr>
<tr>
<td>SCRIP_INH_TEST_RECORD1</td>
<td>scripinhtestrecord1</td>
</tr>
<tr>
<td>SCRIP_INH_TEST_RECORD2</td>
<td>scripinhtestrecord2</td>
</tr>
<tr>
<td>SCRIP_INH_TEST_RECORD3</td>
<td>scripinhtestrecord3</td>
</tr>
<tr>
<td>SCRIP_INH_TEST_RECORD4</td>
<td>scripinhtestrecord4</td>
</tr>
<tr>
<td>SEARCH_CAMPAIGN</td>
<td>searchcampaign</td>
</tr>
<tr>
<td>SEARCH_SCHEDULE</td>
<td>searcheschedule</td>
</tr>
<tr>
<td>SEARCH_URL_TEST_SOURCE_RECORD</td>
<td>searchurltestsourcerecord</td>
</tr>
<tr>
<td>SEARCH_URL_TEST_TARGET_RECORD</td>
<td>searchurltesttargetrecord</td>
</tr>
<tr>
<td>SELECT_OPTIONS_SOURCE_RECORD</td>
<td>selectoptionssourcerecord</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>SERVICE_PURCHASE_ITEM</td>
<td>servicepurchaseitem</td>
</tr>
<tr>
<td>SERVICE_RESALE_ITEM</td>
<td>serviceresaleitem</td>
</tr>
<tr>
<td>SERVICE_SALE_ITEM</td>
<td>servicesaleitem</td>
</tr>
<tr>
<td>SHIPPING_PACKAGE</td>
<td>shippingpackage</td>
</tr>
<tr>
<td>SHIPPING_PARTNERS_PLUGIN</td>
<td>shippingpartnersplugin</td>
</tr>
<tr>
<td>SHIP_ITEM</td>
<td>shipitem</td>
</tr>
<tr>
<td>SHOPPING_CART</td>
<td>shoppingcart</td>
</tr>
<tr>
<td>SIMPLE_RECORD</td>
<td>simplerecord</td>
</tr>
<tr>
<td>SIMPLE_RECORD_LOCATION</td>
<td>simplerecordlocation</td>
</tr>
<tr>
<td>SITE_CATEGORY</td>
<td>sitecategory</td>
</tr>
<tr>
<td>SLAVE</td>
<td>slave</td>
</tr>
<tr>
<td>SLAVE_EMPTY_EXPRESSION</td>
<td>slaveemptyexpression</td>
</tr>
<tr>
<td>SLAVE_FEATURE</td>
<td>slavefeature</td>
</tr>
<tr>
<td>SLAVE_MASTER_PERMISSION</td>
<td>slavemasterpermission</td>
</tr>
<tr>
<td>SLAVE_PERMISSION</td>
<td>slavepermission</td>
</tr>
<tr>
<td>SLAVE_TARGET_PROPERTY</td>
<td>slavetargetproperty</td>
</tr>
<tr>
<td>SLAVE_VALID_EXPRESSION</td>
<td>slavevalidexpression</td>
</tr>
<tr>
<td>SOLUTION</td>
<td>solution</td>
</tr>
<tr>
<td>SORT_BASE_RECORD</td>
<td>sortbaserecord</td>
</tr>
<tr>
<td>SORT_RECORD</td>
<td>sortrecord</td>
</tr>
<tr>
<td>SORTRELATED_RECORD</td>
<td>sortrelatedrecord</td>
</tr>
<tr>
<td>STATIC_LIST_RECORD</td>
<td>staticlistrecord</td>
</tr>
<tr>
<td>STATIC_OPTIONS_FIELD_TEST_RECORD</td>
<td>staticoptionsfieldtestrecord</td>
</tr>
<tr>
<td>STATIC_OPTIONS_VALUE</td>
<td>staticoptionsvalue</td>
</tr>
<tr>
<td>STORE_TAB</td>
<td>storetab</td>
</tr>
<tr>
<td>STUDENT_RECORD</td>
<td>studentrecord</td>
</tr>
<tr>
<td>SUBLIST</td>
<td>sublist</td>
</tr>
<tr>
<td>SUBSIDIARY</td>
<td>subsidiary</td>
</tr>
<tr>
<td>SUBTOTAL_ITEM</td>
<td>subtotalitem</td>
</tr>
<tr>
<td>SUB_SELECT_GROUP_RECORD</td>
<td>subselectgrouprecord</td>
</tr>
<tr>
<td>SUITELET</td>
<td>suitelet</td>
</tr>
<tr>
<td>SUITELET_DEPLOYMENT</td>
<td>suiteletdeployment</td>
</tr>
<tr>
<td>SUITE_SCRIPT_DETAIL</td>
<td>suitescriptdetail</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>---------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>SUPPORT_CASE</td>
<td>supportcase</td>
</tr>
<tr>
<td>SUPPORT_CASE_ISSUE</td>
<td>supportcaseissue</td>
</tr>
<tr>
<td>SUPPORT_CASE_ORIGIN</td>
<td>supportcaseorigin</td>
</tr>
<tr>
<td>SUPPORT_CASE_PRIORITY</td>
<td>supportcasepriority</td>
</tr>
<tr>
<td>SUPPORT_CASE_STATUS</td>
<td>supportcasestatus</td>
</tr>
<tr>
<td>SUPPORT_CASE_TYPE</td>
<td>supportcasetype</td>
</tr>
<tr>
<td>SUPPORT_TERRITORY</td>
<td>supportterritory</td>
</tr>
<tr>
<td>SYSTEM_EMAIL_TEMPLATE</td>
<td>systememailtemplate</td>
</tr>
<tr>
<td>SYSTEM_JOURNAL</td>
<td>systemjournal</td>
</tr>
<tr>
<td>SYSTEM_NOTE</td>
<td>systemnote</td>
</tr>
<tr>
<td>SYSTEM_NOTE_FIELD</td>
<td>systemnotefield</td>
</tr>
<tr>
<td>TABLE_CONDITION_TEST_RECORD</td>
<td>tableconditiontestrecord</td>
</tr>
<tr>
<td>TASK</td>
<td>task</td>
</tr>
<tr>
<td>TASK_ITEM_STATUS</td>
<td>taskitemstatus</td>
</tr>
<tr>
<td>TAX_CALCULATION_PLUGIN</td>
<td>taxcalculationplugin</td>
</tr>
<tr>
<td>TAX_ITEM_TAX_GROUP</td>
<td>taxitemtaxgroup</td>
</tr>
<tr>
<td>TAX_PERIOD</td>
<td>taxperiod</td>
</tr>
<tr>
<td>TAX_TYPE</td>
<td>taxtype</td>
</tr>
<tr>
<td>TERM</td>
<td>term</td>
</tr>
<tr>
<td>TESTDOAGGREGATEDOSUBTYPE</td>
<td>testdoaggregatedosubtype</td>
</tr>
<tr>
<td>TESTDOAGGREGATERESTRICTIONRECORD</td>
<td>testdoaggregaterestrictionrecord</td>
</tr>
<tr>
<td>TEST_COMPOSED_RECORD1</td>
<td>testcomposedrecord1</td>
</tr>
<tr>
<td>TEST_COMPOSED_RECORD2</td>
<td>testcomposedrecord2</td>
</tr>
<tr>
<td>TEST_COMPOSED_RECORD3</td>
<td>testcomposedrecord3</td>
</tr>
<tr>
<td>TESTCONFIGURABLE_RECORD</td>
<td>testconfigurablerecord</td>
</tr>
<tr>
<td>TEST_DO_AGGREGATE_RECORD</td>
<td>testdoaggregaterecord</td>
</tr>
<tr>
<td>TEST_EXPOSURE_RECORD</td>
<td>testexposurerecord</td>
</tr>
<tr>
<td>TEST_FEATURE_RECORD</td>
<td>testfeaturerecord</td>
</tr>
<tr>
<td>TEST_FULL_RECORD</td>
<td>testfullrecord</td>
</tr>
<tr>
<td>TEST_MACROS_UMD_RECORD</td>
<td>testmacrosumdrecord</td>
</tr>
<tr>
<td>TEST_MULTI_TABLE_PERSISTENCE_RECORD</td>
<td>testmultitablepersistencerecord</td>
</tr>
<tr>
<td>TEST_NEW_URLS_RECORD</td>
<td>testnewurlsrecord</td>
</tr>
<tr>
<td>TEST_NEW_URLS_TARGET_RECORD</td>
<td>testnewurlstargetrecord</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>---------------------------------------------------</td>
</tr>
<tr>
<td>TEST_NEW_URLS_UNSUPPORTED_RECORD</td>
<td>testnewurlsunsupportedrecord</td>
</tr>
<tr>
<td>TEST_NEXT_STANDARD_RECORD</td>
<td>testnextstandardrecord</td>
</tr>
<tr>
<td>TEST_PLUGIN</td>
<td>testplugin</td>
</tr>
<tr>
<td>TEST_PRIMARY_RECORD_FOR_RELATIONSHIPS</td>
<td>testprimaryrecordforrelationships</td>
</tr>
<tr>
<td>TEST_RECORD</td>
<td>testrecord</td>
</tr>
<tr>
<td>TEST_RECORD_ACTION_RECORD</td>
<td>testrecordactionrecord</td>
</tr>
<tr>
<td>TEST_RECORD_UNIQUE_KEY</td>
<td>testrecorduniquekey</td>
</tr>
<tr>
<td>TEST_RECORD_WITHOUT_LABEL</td>
<td>testrecordwithoutlabel</td>
</tr>
<tr>
<td>TEST_RECORD_WITH_DISABLED_RECORD_SORT_FIELDS</td>
<td>testrecordwithdisabledrecordsortfields</td>
</tr>
<tr>
<td>TEST_RECORD_WITH_SORT_FIELDS</td>
<td>testrecordwithsortfields</td>
</tr>
<tr>
<td>TEST_REGRESSION_RECORD</td>
<td>testregressionrecord</td>
</tr>
<tr>
<td>TEST_related_PROPERTY</td>
<td>testrelatedproperty</td>
</tr>
<tr>
<td>TEST_SECURED_RECORD</td>
<td>testsecuredrecord</td>
</tr>
<tr>
<td>TEST_SIMPLE_PERSISTENCE_RECORD</td>
<td>testsimplepersistencerecord</td>
</tr>
<tr>
<td>TEST_SIMPLE_PERSISTENCE_SELECT_SIDE_RECORD</td>
<td>testsimplepersistenceselectsiderecord</td>
</tr>
<tr>
<td>TEST_SLAVING_RATE_FIELD_RECORD</td>
<td>testslavingratefieldrecord</td>
</tr>
<tr>
<td>TEST_SLAVING_RECORD</td>
<td>testslavingrecord</td>
</tr>
<tr>
<td>TEST_SLAVING_RECORD_OPTIMIZED</td>
<td>testslavingrecordoptimized</td>
</tr>
<tr>
<td>TEST_SOURCING_MASTER_FIELD_ANNOTATION_MASTER</td>
<td>testsourcingmasterfieldannotationmaster</td>
</tr>
<tr>
<td>TEST_SOURCING_MASTER_FIELD_ANNOTATION_RECORD</td>
<td>testsourcingmasterfieldannotationrecord</td>
</tr>
<tr>
<td>TEST_SOURCING_OPTIONS_CONDITION_MASTER</td>
<td>testsourcingoptionsconditionmaster</td>
</tr>
<tr>
<td>TEST_SOURCING_OPTIONS_CONDITION_RECORD</td>
<td>testsourcingoptionsconditionrecord</td>
</tr>
<tr>
<td>TEST_SOURCING_OPTIONS_CONDITION_TARGET</td>
<td>testsourcingoptionsconditiontarget</td>
</tr>
<tr>
<td>TEST_SOURCING_SUBLIST_FIELD_ANNOTATION_MASTER</td>
<td>testsourcingsublistfieldannotationmaster</td>
</tr>
<tr>
<td>TEST_SOURCING_SUBLIST_FIELD_ANNOTATION_RECORD</td>
<td>testsourcingsublistfieldannotationrecord</td>
</tr>
<tr>
<td>TEST_SOURCING_VALUE_RATE_COL_MASTER</td>
<td>testsourcingvaluateratecolmaster</td>
</tr>
<tr>
<td>TEST_SOURCING_VALUE_RATE_COL_RECORD</td>
<td>testsourcingvaluateratecolrecord</td>
</tr>
<tr>
<td>TEST_STANDARD_RECORD</td>
<td>teststandardrecord</td>
</tr>
<tr>
<td>TEST_TRANSACTION</td>
<td>testtransaction</td>
</tr>
<tr>
<td>TIME_BILL</td>
<td>timebill</td>
</tr>
<tr>
<td>TOPIC</td>
<td>topic</td>
</tr>
<tr>
<td>TRACKING_NUMBER</td>
<td>trackingnumber</td>
</tr>
<tr>
<td>TRANSACTION</td>
<td>transaction</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>-----------------------------------------------------------------------</td>
</tr>
<tr>
<td>TRANSACTION_ADDRESSBOOK</td>
<td>transactionaddressbook</td>
</tr>
<tr>
<td>TRANSACTION_BILLING_ADDRESSBOOK</td>
<td>transactionbillingaddressbook</td>
</tr>
<tr>
<td>TRANSACTION_NUMBERING_AUDIT_LOG</td>
<td>transactionnumberingauditlog</td>
</tr>
<tr>
<td>TRANSACTION_RETURN_ADDRESSBOOK</td>
<td>transactionreturnaddressbook</td>
</tr>
<tr>
<td>TRANSACTION_SHIPPING_ADDRESSBOOK</td>
<td>transactionshippingaddressbook</td>
</tr>
<tr>
<td>TRANSFER</td>
<td>transfer</td>
</tr>
<tr>
<td>TRANSFER_ORDER</td>
<td>transferorder</td>
</tr>
<tr>
<td>TWO_FACTORDEVICE</td>
<td>twofactordevice</td>
</tr>
<tr>
<td>TYPE_FIELD_PARENT_RECORD</td>
<td>typefieldparentrecord</td>
</tr>
<tr>
<td>TYPE_FIELD_RECORD</td>
<td>typefieldrecord</td>
</tr>
<tr>
<td>UMD_FIELD</td>
<td>umdfield</td>
</tr>
<tr>
<td>UNDELIVERED_EMAIL</td>
<td>undeliveredemail</td>
</tr>
<tr>
<td>UNIFICATION_TEST</td>
<td>unificationtest</td>
</tr>
<tr>
<td>USER_EVENT_SCRIPT</td>
<td>usereventscript</td>
</tr>
<tr>
<td>USER_EVENT_SCRIPT_DEPLOYMENT</td>
<td>usereventscriptdeployment</td>
</tr>
<tr>
<td>USRCATEGORY</td>
<td>usrcategory</td>
</tr>
<tr>
<td>USRSAVEDSEARCH</td>
<td>usrsavedsearch</td>
</tr>
<tr>
<td>USR_ANALYTICAL</td>
<td>usranalytical</td>
</tr>
<tr>
<td>USR_AUDITING_SOURCE_RECORD</td>
<td>usrauditingsourcerecord</td>
</tr>
<tr>
<td>USR_AUDIT_LOG</td>
<td>usrauditlog</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_BTH_ROOT</td>
<td>usrchannelagbthroot</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_BTH_ROOT_SUB_TYPE</td>
<td>usrchannelagbthrootsubtype</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_BTH_SEARCH_MTM_ROOT</td>
<td>usrchannelagbthsearchmtmroot</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_BTH_SEARCH_MTM_SUB_TYPE</td>
<td>usrchannelagbthsearchmtmsubtype</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_BTH_SEARCH_MTO_ROOT</td>
<td>usrchannelagbthsearchmtoroot</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_BTH_SEARCH_MTO_SUB_TYPE</td>
<td>usrchannelagbthsearchmtosubtype</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_SRC_ROOT</td>
<td>usrchannelagsrcroot</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_SRC_ROOT_SUB_TYPE</td>
<td>usrchannelagsrcrootsubtype</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_SRC_SEARCH_MTM_PRIMARY</td>
<td>usrchannelagsrcsearchmtmprimary</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_SRC_SEARCH_MTO_PRIMARY</td>
<td>usrchannelagsrcsearchmtoprimary</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_TGT_ROOT</td>
<td>usrchannelagtgtroot</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_TGT_SEARCH_MTM_ROOT</td>
<td>usrchannelagtgtsearchmtmroot</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_TGT_SEARCH_MTM_SUB_TYPE</td>
<td>usrchannelagtgtsearchmtmsubtype</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_TGT_SEARCH_MTO_ROOT</td>
<td>usrchannelagtgtsrchmtoroot</td>
</tr>
<tr>
<td>USR_CHANNEL_AG_TGT_SEARCH_MTO_SUB_TYPE</td>
<td>usrchannelagtgtsrchmtosubtype</td>
</tr>
<tr>
<td>USR_CHANNEL_STD_ROOT</td>
<td>usrchannelstdroot</td>
</tr>
<tr>
<td>USR_CHANNEL_STD_SEARCH_MTM_PRIMARY</td>
<td>usrchannelstdsearchmtmprimary</td>
</tr>
<tr>
<td>USR_CHANNEL_STD_SEARCH_MTO_PRIMARY</td>
<td>usrchannelstdsearchmtoprimary</td>
</tr>
<tr>
<td>USR_EXECUTION_LOG</td>
<td>usrexectionlog</td>
</tr>
<tr>
<td>USR_EXPOSE_EXTERNAL</td>
<td>usrexposeexternal</td>
</tr>
<tr>
<td>USR_EXPOSEIMPORTANT</td>
<td>usrexposeimportant</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_NVAL_MTO_PRIMARY</td>
<td>usrexposeintnlfldplainagbthnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_NVAL_MTO_ROOT</td>
<td>usrexposeintnlfldplainagbthnvalmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_PLAIN_MTO_PRIMARY</td>
<td>usrexposeintnlfldplainagbthplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_PLAIN_MTO_ROOT</td>
<td>usrexposeintnlfldplainagbthplainmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_NVAL_MTO_PRIMARY</td>
<td>usrexposeintnlfldplainagsrcnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_PLAIN_MTO_PRIMARY</td>
<td>usrexposeintnlfldplainagsrcplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_NVAL_MTO_PRIMARY</td>
<td>usrexposeintnlfldplainagtgnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_PRIMARY</td>
<td>usrexposeintnlfldplainagtgtplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_ROOT</td>
<td>usrexposeintnlfldplainagtgtplainmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagbthnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_NVAL_MTO_ROOT</td>
<td>usrexposeintnlfldplainagbthnvalmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagbthplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_PLAIN_MTO_ROOT</td>
<td>usrexposeintnlfldplainagbthplainmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagsrcnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagsrcplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagtgnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagtgtplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_ROOT</td>
<td>usrexposeintnlfldplainagtgtplainmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagsrcnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagsrcplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagtgnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagtgtplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_ROOT</td>
<td>usrexposeintnlfldplainagtgtplainmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagbthnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_NVAL_MTO_ROOT</td>
<td>usrexposeintnlfldplainagbthnvalmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagbthplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_BTH_PLAIN_MTO_ROOT</td>
<td>usrexposeintnlfldplainagbthplainmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagsrcnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_SRC_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagsrcplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_NVAL_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagtgnvalmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO PRIMARY</td>
<td>usrexposeintnlfldplainagtgtplainmtoprimary</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_ROOT</td>
<td>usrexposeintnlfldplainagtgtplainmtoroot</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>USR_EXPOSE_INTNL_FLDPLAIN_AG_TGT_PLAIN_MTO_SUBTYPE</td>
<td>usrexposeintnlfldplainagtgtplainsubtype</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------</td>
</tr>
<tr>
<td>USR_EXPOSEPLAIN_FLDFIELDPLAINAGSRC</td>
<td>usrexposeplainfldplainagsrccroot</td>
</tr>
<tr>
<td>USR_EXPOSEPLAIN_FLDFIELDPLAINAGSRC_SUB</td>
<td>usrexposeplainfldplainagsrcsubtype</td>
</tr>
<tr>
<td>USR_EXPOSEPLAIN_FLDFIELDPLAINAGTGT</td>
<td>usrexposeplainfldplainagtgtroot</td>
</tr>
<tr>
<td>USR_EXPOSEPLAIN_FLDFIELDPLAINAGTGT_SUB</td>
<td>usrexposeplainfldplainagtgtsubtype</td>
</tr>
<tr>
<td>USR_EXPOSEPLAIN_FLDFIELDPLAINSTD</td>
<td>usrexposeplainfldplainstdroot</td>
</tr>
<tr>
<td>USR_EXPOSEPLAIN_FLDFIELDPLAINSTD_SUB</td>
<td>usrexposeplainfldplainstdsubtype</td>
</tr>
<tr>
<td>USR_FEATURE_BTH</td>
<td>usrfeatureagbthroot</td>
</tr>
<tr>
<td>USR_FEATURE_BTH_SUB</td>
<td>usrfeatureagbthrootsubtype</td>
</tr>
<tr>
<td>USR_FEATURE_SRC</td>
<td>usrfeatureagsrcroot</td>
</tr>
<tr>
<td>USR_FEATURE_SRC_SUB</td>
<td>usrfeatureagsrcsubtype</td>
</tr>
<tr>
<td>USR_FEATURE_TGT</td>
<td>usrfeatureagtgtroot</td>
</tr>
<tr>
<td>USR_FEATURE_TGT_SUB</td>
<td>usrfeatureagtgtsubtype</td>
</tr>
<tr>
<td>USR_FEATURE_DEFAULT_COLUMNS_RECORD</td>
<td>usrfeaturecsmdefaultcolumnsrecord</td>
</tr>
<tr>
<td>USR_FEATURE_IMPORTANT_JOIN_RECORD</td>
<td>usrfeaturecsmimportantjoindrecord</td>
</tr>
<tr>
<td>USR_FEATURE_INHERITANCE_RECORD</td>
<td>usrfeaturecsminheritancerecord</td>
</tr>
<tr>
<td>USR_FEATURE_USAGE_SPECIFIC_RECORD</td>
<td>usrfeaturecsmusagespecificrecord</td>
</tr>
<tr>
<td>USR_FEATURE_STD</td>
<td>usrfeaturestdroot</td>
</tr>
<tr>
<td>USR_PERMISSION_BTH_DENIED</td>
<td>usrpsevenagbthdeniedmtmroot</td>
</tr>
<tr>
<td>USR_PERMISSION_BTH_DENIED_SUB</td>
<td>usrpsevenagbthdeniedmtmsubtype</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets Query.type Property To</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>-----------------------------------------------------------------</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_DENIED_MTO_ROOT</td>
<td>usrpbrmgbthdeniedmtoROTO</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_DENIED_MTO_SUBTYPE</td>
<td>usrpbrmgbthdeniedmtosubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_GRANTED_MMTM_ROOT</td>
<td>usrpbrmgbrtmmtmroot</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_GRANTED_MTM_SUBTYPE</td>
<td>usrpbrmgbrtmmtmsubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_GRANTED_MTO_ROOT</td>
<td>usrpbrmgbrtmtoROOT</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_GRANTED_MTO_SUBTYPE</td>
<td>usrpbrmgbrtmtosubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_ROOT</td>
<td>usrpbrmgbrtROOT</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_BTH_ROOT_SUBTYPE</td>
<td>usrpbrmgbrtrootsubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_SRC_DENIED_MTM_PRIMARY</td>
<td>usrpbrmgsrdeniedmtmprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_SRC_DENIED_MTO_PRIMARY</td>
<td>usrpbrmgsrdeniedmtoprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_SRC_GRANTED_MTM_PRIMARY</td>
<td>usrpbrmgsrgrantedmtmprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_SRC_GRANTED_MTO_PRIMARY</td>
<td>usrpbrmgsrgrantedmtoprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_SRC_ROOT</td>
<td>usrpbrmgsrcroot</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_SRC_ROOT_SUBTYPE</td>
<td>usrpbrmgsrcrootsubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_DENIED_MTM_ROOT</td>
<td>usrpbrmggtgtdeniedmtoROOT</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_DENIED_MTM_SUBTYPE</td>
<td>usrpbrmggtgtdeniedmtosubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_DENIED_MTO_ROOT</td>
<td>usrpbrmggtgtdeniedmtoROOT</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_DENIED_MTO_SUBTYPE</td>
<td>usrpbrmggtgtdeniedmtosubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_GRANTED_MTM_ROOT</td>
<td>usrpbrmggtgtgrantedmtoROOT</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_GRANTED_MTM_SUBTYPE</td>
<td>usrpbrmggtgtgrantedmtosubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_GRANTED_MTO_ROOT</td>
<td>usrpbrmggtgtgrantedmtoROOT</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_GRANTED_MTO_SUBTYPE</td>
<td>usrpbrmggtgtgrantedmtosubtype</td>
</tr>
<tr>
<td>USR_PERMISSION_AG_TGT_ROOT</td>
<td>usrpbrmggttROOT</td>
</tr>
<tr>
<td>USR_PERMISSION_STD_DENIED_MTM_PRIMARY</td>
<td>usrpbrmsstddeniedmtmprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_STD_DENIED_MTO_PRIMARY</td>
<td>usrpbrmsstddeniedmtoprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_STD_GRANTED_MTM_PRIMARY</td>
<td>usrpbrmsstdgrantedmtmprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_STD_GRANTED_MTO_PRIMARY</td>
<td>usrpbrmsstdgrantedmtoprimary</td>
</tr>
<tr>
<td>USR_PERMISSION_STD_ROOT</td>
<td>usrpbrmsdROOT</td>
</tr>
<tr>
<td>USR_POLYMORPHIC_CHILD_ONE_RECORD</td>
<td>usrpbrmpcchonerecord</td>
</tr>
<tr>
<td>USR_POLYMORPHIC_CHILD_TWO_RECORD</td>
<td>usrpbrmpcchtworecord</td>
</tr>
<tr>
<td>USR_POLYMORPHIC_JOIN_TEST_RECORD</td>
<td>usrpbrmpcjointestrecord</td>
</tr>
<tr>
<td>USR##_TARGET##_PROPERTIES##_GROUP##_BY##_TARGET_RECORD</td>
<td>usrpbrmsgbytargerrecord</td>
</tr>
<tr>
<td>USR_TARGET_PROPERTIES_MTO2_TARGET_RECORD</td>
<td>usrpbrmsmto2targerrecord</td>
</tr>
</tbody>
</table>
Enum Value | Sets Query.type Property To
---|---
USR_TARGET_PROPERTIES_MTO_TARGET_RECORD | usrtargetpropertiesmtotargetrecord
USR_TARGET_PROPERTIES_ROOT_RECORD | usrtargetpropertiesrootrecord
USR_UNIVERSAL | usruniversal
VENDOR | vendor
VENDOR_BILL | vendorbill
VENDOR_CATEGORY | vendorcategory
VENDOR_CREDIT | vendorcredit
VENDOR_PAYMENT | vendorpayment
VENDOR_SUBSIDIARY_RELATIONSHIP | vendorsubsidiaryrelationship
WEBAPP | webapp
WEB_SITE | website
WIN_LOSS_REASON | winlossreason
WORKFLOW_ACTION_SCRIPT | workflowactionscript
WORKFLOW_ACTION_SCRIPT_DEPLOYMENT | workflowactionscriptdeployment
WORKPLACE | workplace
WORK_CALENDAR | workcalendar

Syntax

```javascript
var search = query.create({
  type: query.Type.CUSTOMER
});
var salesrep = search.autoJoin({
  fieldId: 'salesrep'
});
var cond1 = search.createCondition({
  fieldId: 'id',
  operator: query.Operator.EQUAL,
  values: 107
});
var cond2 = search.createCondition({
  fieldId: 'id',
  operator: query.Operator.EQUAL,
  values: 2647
});
var cond3 = salesrep.createCondition({
  fieldId: 'email',
  operator: query.Operator.START_WITH_NOT,
});
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/query Module Script Samples.
N/query Module

values: 'foo'
});
search.condition = search.and(
    cond3, search.or(cond1, cond2)
);
var resultSet = search.run();

N/record Module

Load the record module to work with NetSuite records. You can use this module to create, delete, copy, load, or make changes to a record.

SuiteScript supports working with standard NetSuite records and with instances of custom record types. Supported standard record types are described in the SuiteScript Records Browser. Refer also to SuiteScript Supported Records. For help working with an instance of a custom record type, see the help topic Custom Record.

For help finding a record's internal ID, see the help topic How do I find a record's internal ID?

Important: SuiteScript does not support direct access to the NetSuite UI through the Document Object Model (DOM). The NetSuite UI should only be accessed with SuiteScript APIs.

- N/record Module Members
- Column Object Members
- Field Object Members
- Macro Object Members
- Record Object Members
- Sublist Object Members
- N/record Module Script Samples
- N/record Default Values

N/record Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>record.Column</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a column of a sublist on a standard or custom record.</td>
</tr>
<tr>
<td></td>
<td>record.Field</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a body or sublist field on a standard or custom record.</td>
</tr>
<tr>
<td></td>
<td>record.Macro</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a NetSuite record macro.</td>
</tr>
<tr>
<td></td>
<td>Plain JavaScript Object</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>A plain JavaScript object of record macros available for a record type. This object is returned by Record.getMacros(options).</td>
</tr>
<tr>
<td></td>
<td>record.Record</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a NetSuite record.</td>
</tr>
<tr>
<td></td>
<td>record.Sublist</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a sublist on a standard or custom record.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
<td>--------------------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Method</td>
<td>record.attach(options)</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Attaches a record to another record.</td>
</tr>
<tr>
<td></td>
<td>record.attach.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Attaches a record asynchronously to another record.</td>
</tr>
<tr>
<td></td>
<td>record.copy(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Creates a new record by copying an existing record in NetSuite.</td>
</tr>
<tr>
<td></td>
<td>record.copy.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Creates a new record asynchronously by copying an existing record in NetSuite.</td>
</tr>
<tr>
<td></td>
<td>record.create(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Creates a new record.</td>
</tr>
<tr>
<td></td>
<td>record.create.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Creates a new record asynchronously.</td>
</tr>
<tr>
<td></td>
<td>record.delete(options)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Deletes a record.</td>
</tr>
<tr>
<td></td>
<td>record.delete.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Deletes a record asynchronously.</td>
</tr>
<tr>
<td></td>
<td>record.detach(options)</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Detaches a record from another record.</td>
</tr>
<tr>
<td></td>
<td>record.detach.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Detaches a record from another record asynchronously.</td>
</tr>
<tr>
<td></td>
<td>record.submitFields.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Updates and submits one or more body fields asynchronously on an existing record in NetSuite, and returns the internal ID of the parent record.</td>
</tr>
<tr>
<td></td>
<td>record.transform(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Transforms a record from one type into another, using data from an existing record.</td>
</tr>
<tr>
<td></td>
<td>record.transform.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Transforms a record from one type into another asynchronously, using data from an existing record.</td>
</tr>
<tr>
<td>Enum</td>
<td>record.Type</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Enumeration that holds the string values for supported standard record types.</td>
</tr>
</tbody>
</table>

### Column Object Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Column.id</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the column.</td>
</tr>
<tr>
<td></td>
<td>Column.label</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the UI label for the column.</td>
</tr>
<tr>
<td></td>
<td>Column.sublistId</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the standard or custom sublist that contains the column.</td>
</tr>
<tr>
<td></td>
<td>Column.type</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the column type.</td>
</tr>
</tbody>
</table>
### Field Object Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Field.#get#Select#Options#()</td>
<td>array</td>
<td>Client and server-side scripts</td>
<td>Returns an array of available options on a standard or custom select, multiselect, or radio field as key-value pairs. Only the first 1,000 available options are returned.</td>
</tr>
<tr>
<td>Property</td>
<td>Field.label</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the UI label for a standard or custom field body or sublist field.</td>
</tr>
<tr>
<td></td>
<td>Field.id</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of a standard or custom body or sublist field.</td>
</tr>
<tr>
<td></td>
<td>Field.type</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the type of a body or sublist field.</td>
</tr>
<tr>
<td></td>
<td>Field.isMandatory</td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the standard or custom field is mandatory on the record form, or false otherwise.</td>
</tr>
<tr>
<td></td>
<td>Field.sublistId</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the ID of the sublist associated with the specified sublist field.</td>
</tr>
<tr>
<td></td>
<td>Field.isDisplay</td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the field is visible on the record form, or false if it is not.</td>
</tr>
</tbody>
</table>

### Macro Object Members

The following members are called on the record.Macro object. For information about record macros, see the help topic Overview of Record Action and Macro APIs.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Macro.execute(options)</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Performs a macro operation and returns its result in an object.</td>
</tr>
<tr>
<td></td>
<td>Macro.execute.#promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Performs a macro operation asynchronously.</td>
</tr>
<tr>
<td></td>
<td>Macro(options)</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Performs a macro operation and returns its result in an object.</td>
</tr>
<tr>
<td></td>
<td>Macro.promise(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Performs a macro operation asynchronously.</td>
</tr>
<tr>
<td>Property</td>
<td>Macro.id</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The ID of the macro. For a list of macro IDs, see the help topic Supported Record Macros.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Macro.label</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The macro label.</td>
</tr>
<tr>
<td></td>
<td>Macro.description</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The macro description.</td>
</tr>
<tr>
<td></td>
<td>Macro.attributes</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>The macro defined attributes.</td>
</tr>
</tbody>
</table>

**Record Object Members**

The following members are called on the `record.Record` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Record.cancelLine</td>
<td>object.Record</td>
<td>Client and server-side scripts</td>
<td>Cancels the currently selected line on a sublist.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.commitLine</td>
<td>object.Record</td>
<td>Client and server-side scripts</td>
<td>Commits the currently selected line on a sublist.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.executeMacro</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Performs macro operation and returns its result in a plain JavaScript object.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.getMacros</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Provides a plain JavaScript object that contains macro objects defined for a record type, indexed by the Macro ID.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.findMatrixLine</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the line number of the first instance where a specified value is found in a specified column of the matrix.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.findSublistLineWithValue</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the line number for the first occurrence of a field value in a sublist.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.getCurrentMatrixSublistValue</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Gets the value for the currently selected line in the matrix.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.getCurrentSublistField</td>
<td>record.Field</td>
<td>Client and server-side scripts</td>
<td>Returns a field object from a sublist.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.getCurrentSublistIndex</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the line number of the currently selected line.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.getCurrentSublistSubrecord</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Gets the subrecord for the associated sublist field on the current line.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.getCurrentSublistText</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Returns a text representation of the field value in the currently selected line.</td>
</tr>
<tr>
<td>Method</td>
<td>Record.getCurrentSublistValue</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the value of a sublist field on the currently selected sublist line.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Record.###getField(options)</td>
<td>record.Field</td>
<td>record.Field</td>
<td>Client and server-side scripts</td>
<td>Returns a field object from a record.</td>
</tr>
<tr>
<td>Record.getFields()</td>
<td>string[]</td>
<td>string[]</td>
<td>Client and server-side scripts</td>
<td>Returns the body field names (internal ids) of all the fields in the record, including machine header field and matrix header fields.</td>
</tr>
<tr>
<td>Record.###getLineCount(options)</td>
<td>number</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the number of lines in a sublist.</td>
</tr>
<tr>
<td>Record.###getMacro(options)</td>
<td>record.Macro</td>
<td>record.Macro</td>
<td>Client and server-side scripts</td>
<td>Provides a macro to execute.</td>
</tr>
<tr>
<td>Record.###getMacros(options)</td>
<td>Object</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Provides a plain JavaScript object that contains macro objects defined for a record type, indexed by the Macro ID.</td>
</tr>
<tr>
<td>Record.###getMatrixHeaderCount(options)</td>
<td>number</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the number of columns for the specified matrix.</td>
</tr>
<tr>
<td>Record.###getMatrixHeaderField(options)</td>
<td>record.Field</td>
<td>record.Field</td>
<td>Client and server-side scripts</td>
<td>Gets the field for the specified header in the matrix.</td>
</tr>
<tr>
<td>Record.###getMatrixHeaderValue(options)</td>
<td>number</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Gets the value for the associated header in the matrix.</td>
</tr>
<tr>
<td>Record.###getMatrixSublistField(options)</td>
<td>record.Field</td>
<td>record.Field</td>
<td>Client and server-side scripts</td>
<td>Gets the field for the specified sublist in the matrix.</td>
</tr>
<tr>
<td>Record.###getMatrixSublistFieldValue(options)</td>
<td>number</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Gets the value for the associated field in the matrix.</td>
</tr>
<tr>
<td>Record.###getSublist(options)</td>
<td>record.Sublist</td>
<td>record.Sublist</td>
<td>Client and server-side scripts</td>
<td>Returns the specified sublist.</td>
</tr>
<tr>
<td>Record.getSublists()</td>
<td>string[]</td>
<td>string[]</td>
<td>Client and server-side scripts</td>
<td>Returns all the names of all the sublists.</td>
</tr>
<tr>
<td>Record.###getSublistField(options)</td>
<td>record.Field</td>
<td>record.Field</td>
<td>Client and server-side scripts</td>
<td>Returns a field object from a sublist.</td>
</tr>
<tr>
<td>Record.###getSublistFields(options)</td>
<td>string[]</td>
<td>string[]</td>
<td>Client and server-side scripts</td>
<td>Returns all the field names in a sublist.</td>
</tr>
<tr>
<td>Record.###getSublistSubrecord(options)</td>
<td>record.Record</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Gets the subrecord associated with a sublist field. (standard mode only)</td>
</tr>
<tr>
<td>Record.###getSublistText(options)</td>
<td>string</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Returns the value of a sublist field in a text representation.</td>
</tr>
<tr>
<td>Record.###getSublistValue(options)</td>
<td>number</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the value of a sublist field.</td>
</tr>
<tr>
<td>Record.###getSubrecord(options)</td>
<td>record.Record</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Gets the subrecord for the associated field.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Record</td>
<td>getText(options)</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Returns the text representation of a field value.</td>
</tr>
<tr>
<td>Record</td>
<td>getValue(options)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns the value of a field.</td>
</tr>
<tr>
<td>Record</td>
<td>hasCurrentSublistSubrecord(options)</td>
<td>boolean</td>
<td>Client and server-side scripts</td>
<td>Returns a value indicating whether the associated sublist field has a subrecord on the current line.</td>
</tr>
<tr>
<td>Record</td>
<td>hasSublistSubrecord(options)</td>
<td>boolean</td>
<td>Client and server-side scripts</td>
<td>Returns a value indicating whether the associated sublist field contains a subrecord.</td>
</tr>
<tr>
<td>Record</td>
<td>hasSubrecord(options)</td>
<td>boolean</td>
<td>Client and server-side scripts</td>
<td>Returns a value indicating whether the field contains a subrecord.</td>
</tr>
<tr>
<td>Record</td>
<td>insertLine(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Inserts a sublist line.</td>
</tr>
<tr>
<td>Record</td>
<td>removeCurrentSublistSubrecord(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Removes the subrecord for the associated sublist field on the current line.</td>
</tr>
<tr>
<td>Record</td>
<td>removeLine(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Removes a sublist line.</td>
</tr>
<tr>
<td>Record</td>
<td>removeSublistSubrecord(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Removes the subrecord for the associated sublist field. (standard mode only)</td>
</tr>
<tr>
<td>Record</td>
<td>removeSubrecord(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Removes the subrecord for the associated field.</td>
</tr>
<tr>
<td>Record</td>
<td>save(options)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Submits a new record or saves edits to an existing record. This method is not available to subrecords.</td>
</tr>
<tr>
<td>Record</td>
<td>save.promise(number)</td>
<td>number</td>
<td>Client scripts</td>
<td>Submits a new record asynchronously or saves edits to an existing record asynchronously. This method is not available to subrecords.</td>
</tr>
<tr>
<td>Record</td>
<td>selectLine(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Selects an existing line in a sublist.</td>
</tr>
<tr>
<td>Record</td>
<td>selectNewLine(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Selects a new line at the end of a sublist.</td>
</tr>
<tr>
<td>Record</td>
<td>setCurrentSublistValue(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value for the line currently selected in the matrix.</td>
</tr>
<tr>
<td>Record</td>
<td>setCurrentSublistText(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value for the field in the currently selected line by a text representation.</td>
</tr>
<tr>
<td>Record</td>
<td>setCurrentValue(options)</td>
<td>Client and server-side scripts</td>
<td>Sets the value for the field in the currently selected line.</td>
<td></td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Method</td>
<td>record.Record.setMatrix.HeaderValue(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value for the associated header in the matrix.</td>
</tr>
<tr>
<td>Method</td>
<td>record.Record.setMatrix.SublistValue(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value for the associated field in the matrix.</td>
</tr>
<tr>
<td>Method</td>
<td>record.Record.setSublistText(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value of a sublist field by a text representation. (standard mode only)</td>
</tr>
<tr>
<td>Method</td>
<td>record.Record.setSublistValue(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value of a sublist field. (standard mode only)</td>
</tr>
<tr>
<td>Method</td>
<td>record.Record.setText(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value of the field by a text representation.</td>
</tr>
<tr>
<td>Method</td>
<td>record.Record.setValue(options)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
<td>Sets the value of a field.</td>
</tr>
<tr>
<td>Property</td>
<td>record.Record.id</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>The internal ID of a specific record. This property is not available to subrecords.</td>
</tr>
<tr>
<td>Property</td>
<td>record.Record.isDynamic</td>
<td>boolean (read-only)</td>
<td>Client and server-side scripts</td>
<td>Indicates whether the record is in dynamic or standard mode.</td>
</tr>
<tr>
<td>Property</td>
<td>record.Record.type</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>The record type. This property is not available to subrecords.</td>
</tr>
</tbody>
</table>

**Sublist Object Members**

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>sublist.getColumn(options)</td>
<td>sublist.Column</td>
<td>Client and server-side scripts</td>
<td>Returns a column in the sublist.</td>
</tr>
<tr>
<td>Property</td>
<td>sublist.id</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the sublist.</td>
</tr>
<tr>
<td>Property</td>
<td>sublist.isChanged</td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Indicates whether the sublist has changed on the record form.</td>
</tr>
<tr>
<td>Property</td>
<td>sublist.isDisplay</td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Indicates whether the sublist is displayed on the record form.</td>
</tr>
<tr>
<td>Property</td>
<td>sublist.type</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the sublist type.</td>
</tr>
</tbody>
</table>

**N/record Module Script Samples**

The following script samples demonstrate how to use the record module.

These samples use the `require` function, so that you can copy each script into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For more information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.
The following example shows how to create and save a contact record.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/record'], function(record) {
    function createAndSaveContactRecord() {
        var nameData = {
            firstname: 'John',
            middlename: 'Doe',
            lastname: 'Smith'
        };
        var objRecord = record.create({
            type: record.Type.CONTACT,
            isDynamic: true
        });
        objRecord.setValue({'
            fieldId: 'subsidiary',
            value: '1'
        });
        for ( var key in nameData) {
            if (nameData.hasOwnProperty(key)) {
                objRecord.setValue({'
                    fieldId: key,
                    value: nameData[key]
                });
            }
        }
        var recordId = objRecord.save({
            enableSourcing: false,
            ignoreMandatoryFields: false
        });
        createAndSaveContactRecord();
    }
});
createAndSaveContactRecord();
```

The following example shows how to create and save a contact record using Promise methods.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/record'], function(record) {
    function createAndSaveContactRecordWithPromise() {
        var nameData = {
            firstname: 'John',
            middlename: 'Doe',
            lastname: 'Smith'
        };
        var objRecord = record.create({
            type: record.Type.CONTACT,
            isDynamic: true
        });
        objRecord.setValue({'
            fieldId: 'subsidiary',
            value: '1'
        });
        var recordId = objRecord.save({
            enableSourcing: false,
            ignoreMandatoryFields: false
        });
        createAndSaveContactRecordWithPromise();
    }
});
createAndSaveContactRecordWithPromise();
```

Important: Some of the values in these samples are placeholders. Before using these samples, replace all hardcoded values, such as IDs and file paths, with valid values from your NetSuite account. If you run a script with an invalid value, the system may throw an error.

Note: To debug client scripts like the following, we recommend that you use Chrome DevTools for Chrome, Firebug debugger for Firefox, or Microsoft Script Debugger for Internet Explorer. For information about these tools, see the documentation provided with each browser.
var nameData = {
    firstname: 'John',
    middlename: 'Doe',
    lastname: 'Smith'
};
var createRecordPromise = record.create.promise(
    type: record.Type.CONTACT,
    isDynamic: true
);
createRecordPromise.then(function(objRecord) {
    console.log('start evaluating promise content');
    objRecord.setValue(
        fieldId: 'subsidiary',
        value: '1'
    );
    for (var key in nameData) {
        if (nameData.hasOwnProperty(key)) {
            objRecord.setValue(
                fieldId: key,
                value: nameData[key]
            );
        }
    }
    var recordId = objRecord.save(
        enableSourcing: false,
        ignoreMandatoryFields: false
    );
}, function(e) {
    log.error('Unable to create contact', e.name);
});
createAndSaveContactRecordWithPromise();

The following example shows how to access sublists and a subrecord from a record. This example requires the Advanced Number Inventory Management feature.

```javascript
/**
 * @NApiVersion 2.x
 *
 * require(['N/record'], function(record) {
 *     function createPurchaseOrder() {
 *         var rec = record.create(
 *             type: 'purchaseorder',
 *             isDynamic: true
 *         );
 *         rec.setValue(
 *             fieldId: 'entity',
 *             value: 52
 *         );
 *         rec.setValue(
 *             fieldId: 'location',
 *             value: 2
 *         );
 *     }
 *     createAndSavePurchaseOrderWithPromise();
 * });
 */
```
rec.selectNewLine({
    sublistId: 'item'
});
rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: 190
});
rec.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'quantity',
    value: 2
});
subrecordInvDetail = rec.getCurrentSublistSubrecord({
    sublistId: 'item',
    fieldId: 'inventorydetail'
});
subrecordInvDetail.selectNewLine({
    sublistId: 'inventoryassignment'
});
subrecordInvDetail.setCurrentSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber',
    value: 'myinventoryNumber'
});
subrecordInvDetail.commitLine({
    sublistId: 'inventoryassignment'
});
subrecordInvDetail.selectLine({
    sublistId: 'inventoryassignment',
    line: 0
});
var myInventoryNumber = subrecordInvDetail.getCurrentSublistValue({
    sublistId: 'inventoryassignment',
    fieldId: 'receiptinventorynumber'
});
rec.commitLine({
    sublistId: 'item'
});
var recordId = rec.save();
}
createPurchaseOrder();
function createPurchaseOrder() {
    var createRecordPromise = record.create.promise({
        type: 'purchaseorder',
        isDynamic: true
    });
    createRecordPromise.then(function(rec) {
        rec.setValue({
            fieldId: 'entity',
            value: 52
        });
        rec.setValue({
            fieldId: 'location',
            value: 2
        });
        rec.selectNewLine({
            sublistId: 'item'
        });
        rec.setCurrentSublistValue({
            sublistId: 'item',
            fieldId: 'item',
            value: 190
        });
        rec.setCurrentSublistValue({
            sublistId: 'item',
            fieldId: 'quantity',
            value: 2
        });
        subrecordInvDetail = rec.getCurrentSublistSubrecord({
            sublistId: 'item',
            fieldId: 'inventorydetail'
        });
        subrecordInvDetail.selectNewLine({
            sublistId: 'inventoryassignment'
        });
        subrecordInvDetail.setCurrentSublistValue({
            sublistId: 'inventoryassignment',
            fieldId: 'receiptinventorynumber',
            value: 'myinventoryNumber'
        });
        subrecordInvDetail.commitLine({
            sublistId: 'inventoryassignment'
        });
        subrecordInvDetail.selectLine({
            sublistId: 'inventoryassignment',
            line: 0
        });
        var myInventoryNumber = subrecordInvDetail.getCurrentSublistValue({
            sublistId: 'inventoryassignment',
            fieldId: 'receiptinventorynumber'
        });
        rec.commitLine({
            sublistId: 'item'
        });
    });
}
The following example shows you how to call a calculateTax macro on a sales order record. To execute a macro on a record, the record must be created or loaded in dynamic mode. Note that the SuiteTax feature must be enabled to successfully execute the macro used in this sample.

For information about record macros, see the help topic [Overview of Record Action and Macro APIs](#).

```javascript
/**
  * @NApiVersion 2.x
  */
require(['N/record'],
  function(record) {
    var recordObj = record.create({
        type: record.Type.SALES_ORDER,
        isDynamic: true
    });

    var ENTITY_VALUE = 1;
    var ITEM_VALUE = 1;
    recordObj.setValue({
        fieldId: 'entity',
        value: ENTITY_VALUE
    });
    recordObj.selectNewLine({
        sublistId: 'item'
    });
    recordObj.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'item',
        value: ITEM_VALUE
    });
    recordObj.setCurrentSublistValue({
        sublistId: 'item',
        fieldId: 'quantity',
        value: 1
    });
    recordObj.commitLine({
        sublistId:'item'
    });

    var totalBeforeTax = recordObj.getValue({fieldId: 'total'});

    // get macros available on the record
    var macros = recordObj.getMacros();

    // execute the macro
    if ('calculateTax' in macros)
    {
```
N/record Module

```javascript
macros.calculateTax(); // For promise version use: macros.calculateTax.promise()

// Alternative (direct) macro execution
var calculateTax = recordObj.getMacro({id: 'calculateTax'});
// calculateTax(); // For promise version use: calculateTax.promise()
var totalAfterTax = recordObj.getValue({fieldId: 'total'});

var recordId = recordObj.save({
    enableSourcing: false,
    ignoreMandatoryFields: false
});
```

N/record Default Values

You can use SuiteScript 2.0 to specify record initialization parameters that default when creating, copying, loading, and transforming records. To enable this behavior, use the optional `defaultValues` parameter in the following APIs:

- `record.create(options)`
- `record.copy(options)`
- `record.transform(options)`
- `record.load(options)`

The following table lists initialization types that are available to certain SuiteScript-supported records and the values they can contain.

<table>
<thead>
<tr>
<th>Record</th>
<th>Initialization Type</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>All SuiteScript-supported records that support form customization.</td>
<td>customform</td>
<td><code>&lt;customformid&gt;</code></td>
</tr>
<tr>
<td>Assembly Build</td>
<td>assemblyitem</td>
<td><code>&lt;assemblyitemid&gt;</code></td>
</tr>
<tr>
<td>Cash Refund</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Cash Sale</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Check</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Credit Memo</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Customer Payment</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Customer Refund</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Deposit</td>
<td>disablepaymentfilters</td>
<td><code>&lt;disablepaymentfilters&gt;</code></td>
</tr>
<tr>
<td>Estimate</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Expense Report</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Invoice</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Item Receipt</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Non-Inventory Part</td>
<td>subtype</td>
<td>sale</td>
</tr>
<tr>
<td>Opportunity</td>
<td>entity</td>
<td><code>&lt;entityid&gt;</code></td>
</tr>
<tr>
<td>Record</td>
<td>Initialization Type</td>
<td>Values</td>
</tr>
<tr>
<td>------------------------</td>
<td>---------------------</td>
<td>----------------------------------</td>
</tr>
<tr>
<td>Other Charge Item</td>
<td>subtype</td>
<td>sale</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>entity</td>
<td>&lt;entityid&gt;</td>
</tr>
<tr>
<td>Return Authorization</td>
<td>entity</td>
<td>&lt;entityid&gt;</td>
</tr>
<tr>
<td>Sales Order</td>
<td>entity</td>
<td>&lt;entityid&gt;</td>
</tr>
<tr>
<td>Script Deployment</td>
<td>script</td>
<td>&lt;scriptid&gt;</td>
</tr>
<tr>
<td>Service</td>
<td>subtype</td>
<td>sale</td>
</tr>
<tr>
<td>Tax Group</td>
<td>nexuscountry</td>
<td>&lt;countrycode&gt;</td>
</tr>
<tr>
<td>Tax Type</td>
<td>country</td>
<td>&lt;countrycode&gt;</td>
</tr>
<tr>
<td>Topic</td>
<td>parenttopic</td>
<td>&lt;parenttopicid&gt;</td>
</tr>
<tr>
<td>Vendor Bill</td>
<td>entity</td>
<td>&lt;entityid&gt;</td>
</tr>
<tr>
<td>Vendor Payment</td>
<td>entity</td>
<td>&lt;entityid&gt;</td>
</tr>
<tr>
<td>Work Order</td>
<td>assemblyitem</td>
<td>&lt;assemblyitemid&gt;</td>
</tr>
</tbody>
</table>

### Country Codes Used for Initialization Parameters

If you are scripting the Tax Group or Tax Type records, you can initialize the record to source all values related to a specific country. In your script, use the country code for the `countrycodeid` value, for example:

```javascript
record.create('taxgroup', {nexuscountry: 'AR'});
```

<table>
<thead>
<tr>
<th>Country Code</th>
<th>Country Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD</td>
<td>Andorra</td>
</tr>
<tr>
<td>AE</td>
<td>United Arab Emirates</td>
</tr>
<tr>
<td>AF</td>
<td>Afghanistan</td>
</tr>
<tr>
<td>AG</td>
<td>Antigua and Barbuda</td>
</tr>
<tr>
<td>AI</td>
<td>Anguilla</td>
</tr>
<tr>
<td>AL</td>
<td>Albania</td>
</tr>
<tr>
<td>AM</td>
<td>Armenia</td>
</tr>
<tr>
<td>AO</td>
<td>Angola</td>
</tr>
<tr>
<td>AQ</td>
<td>Antarctica</td>
</tr>
<tr>
<td>AR</td>
<td>Argentina</td>
</tr>
<tr>
<td>Country Code</td>
<td>Country Name</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>AS</td>
<td>American Samoa</td>
</tr>
<tr>
<td>AT</td>
<td>Austria</td>
</tr>
<tr>
<td>AU</td>
<td>Australia</td>
</tr>
<tr>
<td>AW</td>
<td>Aruba</td>
</tr>
<tr>
<td>AX</td>
<td>Aland Islands</td>
</tr>
<tr>
<td>AZ</td>
<td>Azerbaijan</td>
</tr>
<tr>
<td>BA</td>
<td>Bosnia and Herzegovina</td>
</tr>
<tr>
<td>BB</td>
<td>Barbados</td>
</tr>
<tr>
<td>BD</td>
<td>Bangladesh</td>
</tr>
<tr>
<td>BE</td>
<td>Belgium</td>
</tr>
<tr>
<td>BF</td>
<td>Burkina Faso</td>
</tr>
<tr>
<td>BG</td>
<td>Bulgaria</td>
</tr>
<tr>
<td>BH</td>
<td>Bahrain</td>
</tr>
<tr>
<td>BI</td>
<td>Burundi</td>
</tr>
<tr>
<td>BJ</td>
<td>Benin</td>
</tr>
<tr>
<td>BL</td>
<td>Saint Barthélemy</td>
</tr>
<tr>
<td>BM</td>
<td>Bermuda</td>
</tr>
<tr>
<td>BN</td>
<td>Brunei Darrussalam</td>
</tr>
<tr>
<td>BO</td>
<td>Bolivia</td>
</tr>
<tr>
<td>BQ</td>
<td>Bonaire, Saint Eustatius, and Saba</td>
</tr>
<tr>
<td>BR</td>
<td>Brazil</td>
</tr>
<tr>
<td>BS</td>
<td>Bahamas</td>
</tr>
<tr>
<td>BT</td>
<td>Bhutan</td>
</tr>
<tr>
<td>BV</td>
<td>Bouvet Island</td>
</tr>
<tr>
<td>BW</td>
<td>Botswana</td>
</tr>
<tr>
<td>BY</td>
<td>Belarus</td>
</tr>
<tr>
<td>BZ</td>
<td>Belize</td>
</tr>
<tr>
<td>CA</td>
<td>Canada</td>
</tr>
<tr>
<td>CC</td>
<td>Cocos (Keeling) Islands</td>
</tr>
<tr>
<td>CD</td>
<td>Congo, Democratic People's Republic</td>
</tr>
<tr>
<td>CF</td>
<td>Central African Republic</td>
</tr>
<tr>
<td>CG</td>
<td>Congo, Republic of</td>
</tr>
<tr>
<td>Country Code</td>
<td>Country Name</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------------</td>
</tr>
<tr>
<td>CH</td>
<td>Switzerland</td>
</tr>
<tr>
<td>CI</td>
<td>Cote d'Ivoire</td>
</tr>
<tr>
<td>CK</td>
<td>Cook Islands</td>
</tr>
<tr>
<td>CL</td>
<td>Chile</td>
</tr>
<tr>
<td>CM</td>
<td>Cameroon</td>
</tr>
<tr>
<td>CN</td>
<td>China</td>
</tr>
<tr>
<td>CO</td>
<td>Colombia</td>
</tr>
<tr>
<td>CR</td>
<td>Costa Rica</td>
</tr>
<tr>
<td>CU</td>
<td>Cuba</td>
</tr>
<tr>
<td>CV</td>
<td>Cape Verde</td>
</tr>
<tr>
<td>CW</td>
<td>Curacao</td>
</tr>
<tr>
<td>CX</td>
<td>Christmas Island</td>
</tr>
<tr>
<td>CY</td>
<td>Cyprus</td>
</tr>
<tr>
<td>CZ</td>
<td>Czech Republic</td>
</tr>
<tr>
<td>DE</td>
<td>Germany</td>
</tr>
<tr>
<td>DJ</td>
<td>Djibouti</td>
</tr>
<tr>
<td>DK</td>
<td>Denmark</td>
</tr>
<tr>
<td>DM</td>
<td>Dominica</td>
</tr>
<tr>
<td>DO</td>
<td>Dominican Republic</td>
</tr>
<tr>
<td>DZ</td>
<td>Algeria</td>
</tr>
<tr>
<td>EA</td>
<td>Ceuta and Melilla</td>
</tr>
<tr>
<td>EC</td>
<td>Ecuador</td>
</tr>
<tr>
<td>EE</td>
<td>Estonia</td>
</tr>
<tr>
<td>EG</td>
<td>Egypt</td>
</tr>
<tr>
<td>EH</td>
<td>Western Sahara</td>
</tr>
<tr>
<td>ER</td>
<td>Eritrea</td>
</tr>
<tr>
<td>ES</td>
<td>Spain</td>
</tr>
<tr>
<td>ET</td>
<td>Ethiopia</td>
</tr>
<tr>
<td>FI</td>
<td>Finland</td>
</tr>
<tr>
<td>FJ</td>
<td>Fiji</td>
</tr>
<tr>
<td>FK</td>
<td>Falkland Islands</td>
</tr>
<tr>
<td>FM</td>
<td>Micronesia, Federal State of...</td>
</tr>
<tr>
<td>Country Code</td>
<td>Country Name</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>FO</td>
<td>Faroe Islands</td>
</tr>
<tr>
<td>FR</td>
<td>France</td>
</tr>
<tr>
<td>GA</td>
<td>Gabon</td>
</tr>
<tr>
<td>GB</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>GD</td>
<td>Grenada</td>
</tr>
<tr>
<td>GE</td>
<td>Georgia</td>
</tr>
<tr>
<td>GF</td>
<td>French Guiana</td>
</tr>
<tr>
<td>GG</td>
<td>Guernsey</td>
</tr>
<tr>
<td>GH</td>
<td>Ghana</td>
</tr>
<tr>
<td>GI</td>
<td>Gibraltar</td>
</tr>
<tr>
<td>GL</td>
<td>Greenland</td>
</tr>
<tr>
<td>GM</td>
<td>Gambia</td>
</tr>
<tr>
<td>GN</td>
<td>Guinea</td>
</tr>
<tr>
<td>GP</td>
<td>Guadeloupe</td>
</tr>
<tr>
<td>GQ</td>
<td>Equatorial Guinea</td>
</tr>
<tr>
<td>GR</td>
<td>Greece</td>
</tr>
<tr>
<td>GS</td>
<td>South Georgia</td>
</tr>
<tr>
<td>GT</td>
<td>Guatemala</td>
</tr>
<tr>
<td>GU</td>
<td>Guam</td>
</tr>
<tr>
<td>GW</td>
<td>Guinea-Bissau</td>
</tr>
<tr>
<td>GY</td>
<td>Guyana</td>
</tr>
<tr>
<td>HK</td>
<td>Hong Kong</td>
</tr>
<tr>
<td>HM</td>
<td>Heard and McDonald Islands</td>
</tr>
<tr>
<td>HN</td>
<td>Honduras</td>
</tr>
<tr>
<td>HR</td>
<td>Croatia/Hrvatska</td>
</tr>
<tr>
<td>HT</td>
<td>Haiti</td>
</tr>
<tr>
<td>HU</td>
<td>Hungary</td>
</tr>
<tr>
<td>IC</td>
<td>Canary Islands</td>
</tr>
<tr>
<td>ID</td>
<td>Indonesia</td>
</tr>
<tr>
<td>IE</td>
<td>Ireland</td>
</tr>
<tr>
<td>IL</td>
<td>Israel</td>
</tr>
<tr>
<td>IM</td>
<td>Isle of Man</td>
</tr>
<tr>
<td>Country Code</td>
<td>Country Name</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>IN</td>
<td>India</td>
</tr>
<tr>
<td>IO</td>
<td>British Indian Ocean Territory</td>
</tr>
<tr>
<td>IQ</td>
<td>Iraq</td>
</tr>
<tr>
<td>IR</td>
<td>Iran (Islamic Republic of)</td>
</tr>
<tr>
<td>IS</td>
<td>Iceland</td>
</tr>
<tr>
<td>IT</td>
<td>Italy</td>
</tr>
<tr>
<td>JE</td>
<td>Jersey</td>
</tr>
<tr>
<td>JM</td>
<td>Jamaica</td>
</tr>
<tr>
<td>JO</td>
<td>Jordan</td>
</tr>
<tr>
<td>JP</td>
<td>Japan</td>
</tr>
<tr>
<td>KE</td>
<td>Kenya</td>
</tr>
<tr>
<td>KG</td>
<td>Kyrgyzstan</td>
</tr>
<tr>
<td>KH</td>
<td>Cambodia</td>
</tr>
<tr>
<td>KI</td>
<td>Kiribati</td>
</tr>
<tr>
<td>KM</td>
<td>Comoros</td>
</tr>
<tr>
<td>KN</td>
<td>Saint Kitts and Nevis</td>
</tr>
<tr>
<td>KP</td>
<td>Korea, Democratic People's Republic</td>
</tr>
<tr>
<td>KR</td>
<td>Korea, Republic of</td>
</tr>
<tr>
<td>KW</td>
<td>Kuwait</td>
</tr>
<tr>
<td>KY</td>
<td>Cayman Islands</td>
</tr>
<tr>
<td>KZ</td>
<td>Kazakhstan</td>
</tr>
<tr>
<td>LA</td>
<td>Lao People's Democratic Republic</td>
</tr>
<tr>
<td>LB</td>
<td>Lebanon</td>
</tr>
<tr>
<td>LC</td>
<td>Saint Lucia</td>
</tr>
<tr>
<td>LI</td>
<td>Liechtenstein</td>
</tr>
<tr>
<td>LK</td>
<td>Sri Lanka</td>
</tr>
<tr>
<td>LR</td>
<td>Liberia</td>
</tr>
<tr>
<td>LS</td>
<td>Lesotho</td>
</tr>
<tr>
<td>LT</td>
<td>Lithuania</td>
</tr>
<tr>
<td>LU</td>
<td>Luxembourg</td>
</tr>
<tr>
<td>LV</td>
<td>Latvia</td>
</tr>
<tr>
<td>LY</td>
<td>Libya</td>
</tr>
<tr>
<td>Country Code</td>
<td>Country Name</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>MA</td>
<td>Morocco</td>
</tr>
<tr>
<td>MC</td>
<td>Monaco</td>
</tr>
<tr>
<td>MD</td>
<td>Moldova, Republic of</td>
</tr>
<tr>
<td>ME</td>
<td>Montenegro</td>
</tr>
<tr>
<td>MF</td>
<td>Saint Martin</td>
</tr>
<tr>
<td>MG</td>
<td>Madagascar</td>
</tr>
<tr>
<td>MH</td>
<td>Marshall Islands</td>
</tr>
<tr>
<td>MK</td>
<td>Macedonia</td>
</tr>
<tr>
<td>ML</td>
<td>Mali</td>
</tr>
<tr>
<td>MM</td>
<td>Myanmar</td>
</tr>
<tr>
<td>MN</td>
<td>Mongolia</td>
</tr>
<tr>
<td>MO</td>
<td>Macau</td>
</tr>
<tr>
<td>MP</td>
<td>Northern Mariana Islands</td>
</tr>
<tr>
<td>MQ</td>
<td>Martinique</td>
</tr>
<tr>
<td>MR</td>
<td>Mauritania</td>
</tr>
<tr>
<td>MS</td>
<td>Montserrat</td>
</tr>
<tr>
<td>MT</td>
<td>Malta</td>
</tr>
<tr>
<td>MU</td>
<td>Mauritius</td>
</tr>
<tr>
<td>MV</td>
<td>Maldives</td>
</tr>
<tr>
<td>MW</td>
<td>Malawi</td>
</tr>
<tr>
<td>MX</td>
<td>Mexico</td>
</tr>
<tr>
<td>MY</td>
<td>Malaysia</td>
</tr>
<tr>
<td>MZ</td>
<td>Mozambique</td>
</tr>
<tr>
<td>NA</td>
<td>Namibia</td>
</tr>
<tr>
<td>NC</td>
<td>New Caledonia</td>
</tr>
<tr>
<td>NE</td>
<td>Niger</td>
</tr>
<tr>
<td>NF</td>
<td>Norfolk Island</td>
</tr>
<tr>
<td>NG</td>
<td>Nigeria</td>
</tr>
<tr>
<td>NI</td>
<td>Nicaragua</td>
</tr>
<tr>
<td>NL</td>
<td>Netherlands</td>
</tr>
<tr>
<td>NO</td>
<td>Norway</td>
</tr>
<tr>
<td>NP</td>
<td>Nepal</td>
</tr>
<tr>
<td>Country Code</td>
<td>Country Name</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>NR</td>
<td>Nauru</td>
</tr>
<tr>
<td>NU</td>
<td>Niue</td>
</tr>
<tr>
<td>NZ</td>
<td>New Zealand</td>
</tr>
<tr>
<td>OM</td>
<td>Oman</td>
</tr>
<tr>
<td>PA</td>
<td>Panama</td>
</tr>
<tr>
<td>PE</td>
<td>Peru</td>
</tr>
<tr>
<td>PF</td>
<td>French Polynesia</td>
</tr>
<tr>
<td>PG</td>
<td>Papua New Guinea</td>
</tr>
<tr>
<td>PH</td>
<td>Philippines</td>
</tr>
<tr>
<td>PK</td>
<td>Pakistan</td>
</tr>
<tr>
<td>PL</td>
<td>Poland</td>
</tr>
<tr>
<td>PM</td>
<td>St. Pierre and Miquelon</td>
</tr>
<tr>
<td>PN</td>
<td>Pitcairn Island</td>
</tr>
<tr>
<td>PR</td>
<td>Puerto Rico</td>
</tr>
<tr>
<td>PS</td>
<td>State of Palestine</td>
</tr>
<tr>
<td>PT</td>
<td>Portugal</td>
</tr>
<tr>
<td>PW</td>
<td>Palau</td>
</tr>
<tr>
<td>PY</td>
<td>Paraguay</td>
</tr>
<tr>
<td>QA</td>
<td>Qatar</td>
</tr>
<tr>
<td>RE</td>
<td>Reunion Island</td>
</tr>
<tr>
<td>RO</td>
<td>Romania</td>
</tr>
<tr>
<td>RS</td>
<td>Serbia</td>
</tr>
<tr>
<td>RU</td>
<td>Russian Federation</td>
</tr>
<tr>
<td>RW</td>
<td>Rwanda</td>
</tr>
<tr>
<td>SA</td>
<td>Saudi Arabia</td>
</tr>
<tr>
<td>SB</td>
<td>Solomon Islands</td>
</tr>
<tr>
<td>SC</td>
<td>Seychelles</td>
</tr>
<tr>
<td>SD</td>
<td>Sudan</td>
</tr>
<tr>
<td>SE</td>
<td>Sweden</td>
</tr>
<tr>
<td>SG</td>
<td>Singapore</td>
</tr>
<tr>
<td>SH</td>
<td>Saint Helena</td>
</tr>
<tr>
<td>SI</td>
<td>Slovenia</td>
</tr>
<tr>
<td>Country Code</td>
<td>Country Name</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>SJ</td>
<td>Svalbard and Jan Mayen Islands</td>
</tr>
<tr>
<td>SK</td>
<td>Slovak Republic</td>
</tr>
<tr>
<td>SL</td>
<td>Sierra Leone</td>
</tr>
<tr>
<td>SM</td>
<td>San Marino</td>
</tr>
<tr>
<td>SN</td>
<td>Senegal</td>
</tr>
<tr>
<td>SO</td>
<td>Somalia</td>
</tr>
<tr>
<td>SR</td>
<td>Suriname</td>
</tr>
<tr>
<td>SS</td>
<td>South Sudan</td>
</tr>
<tr>
<td>ST</td>
<td>Sao Tome and Principe</td>
</tr>
<tr>
<td>SV</td>
<td>El Salvador</td>
</tr>
<tr>
<td>SX</td>
<td>Sint Maarten</td>
</tr>
<tr>
<td>SY</td>
<td>Syrian Arab Republic</td>
</tr>
<tr>
<td>SZ</td>
<td>Swaziland</td>
</tr>
<tr>
<td>TC</td>
<td>Turks and Caicos Islands</td>
</tr>
<tr>
<td>TD</td>
<td>Chad</td>
</tr>
<tr>
<td>TF</td>
<td>French Southern Territories</td>
</tr>
<tr>
<td>TG</td>
<td>Togo</td>
</tr>
<tr>
<td>TH</td>
<td>Thailand</td>
</tr>
<tr>
<td>TJ</td>
<td>Tajikistan</td>
</tr>
<tr>
<td>TK</td>
<td>Tokelau</td>
</tr>
<tr>
<td>TM</td>
<td>Turkmenistan</td>
</tr>
<tr>
<td>TN</td>
<td>Tunisia</td>
</tr>
<tr>
<td>TO</td>
<td>Tonga</td>
</tr>
<tr>
<td>TP</td>
<td>East Timor</td>
</tr>
<tr>
<td>TR</td>
<td>Turkey</td>
</tr>
<tr>
<td>TT</td>
<td>Trinidad and Tobago</td>
</tr>
<tr>
<td>TV</td>
<td>Tuvalu</td>
</tr>
<tr>
<td>TW</td>
<td>Taiwan</td>
</tr>
<tr>
<td>TZ</td>
<td>Tanzania</td>
</tr>
<tr>
<td>UA</td>
<td>Ukraine</td>
</tr>
<tr>
<td>UG</td>
<td>Uganda</td>
</tr>
<tr>
<td>UM</td>
<td>US Minor Outlying Islands</td>
</tr>
</tbody>
</table>
Country Code | Country Name
--- | ---
US | United States
UY | Uruguay
UZ | Uzbekistan
VA | Holy See (City Vatican State)
VC | Saint Vincent and the Grenadines
VE | Venezuela
VG | Virgin Islands (British)
VI | Virgin Islands (USA)
VN | Vietnam
VU | Vanuatu
WF | Wallis and Futuna Islands
WS | Samoa
XK | Kosovo
YE | Yemen
YT | Mayotte
ZA | South Africa
ZM | Zambia
ZW | Zimbabwe

### record.Column

**Object Description**

Encapsulates a column of a sublist on a standard or custom record.

For a complete list of this object’s properties, see [Column Object Members](#).

This object does not return a value, it returns information about the sublist column.

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/record Module

**Since**

2015.2

**Syntax**

⚠️ **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
```
... var objRecord = record.load(
  type: record.Type.SALES_ORDER,
  id: 275
));

var objSublist = objRecord.getSublist({
  sublistId: 'item'
});

var objColumn = objSublist.getColumn({
  fieldId: 'item'
});

if(objColumn.label === 'myLabel '){
  //Perform an action
}
if(objColumn.type === 'checkbox '){
  //Perform an action
}
...
// Add additional code.

---

**Column.id**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the internal ID of the column. Note that the Column.id value is the same as the value that is passed into fieldID.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>string (read-only)</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client and server-side scripts&lt;br&gt;For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/record Module</td>
</tr>
<tr>
<td><strong>Sibling Object Members</strong></td>
<td>Column Object Members</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
  type: record.Type.SALES_ORDER,
  id: 275
});

var objSublist = objRecord.getSublist({
  sublistId: 'item'
});
```

---
### Column.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the internal ID of the column.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This property does not return a value, it returns information about the column label.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/record Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Sibling Object Members</th>
<th>Column Object Members</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2015.2</th>
</tr>
</thead>
</table>

### Syntax

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

var objColumn = objSublist.getColumn({
    fieldId: 'item'
});

if(objColumn.label === 'myLabel') {
    //Perform an action
}...
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.
Column.sublistId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the internal ID of the standard or custom sublist that contains the column.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts&lt;br&gt;For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Column Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

var objColumn = objSublist.getColumn({
    fieldId: 'item'
});

// Perform an action with the objColumn.sublistId value
...
// Add additional code.
```

Column.type

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the column type. For more information on possible return values, see format.Type.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts&lt;br&gt;For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>
record.Field

Object Description
Encapsulates a body or sublist field on a standard or custom record.

Use the following methods to access the Field object:

- `Record.getField(options)`
- `Record.getSublistField(options)`
- `Record.getCurrentSublistField(options)`
- `CurrentRecord.getField(options)`
- `CurrentRecord.getSublistField(options)`

For a complete list of this object's methods and properties, see Field Object Members.

Supported Script Types
Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

var objField = objSublist.getField({
    fieldId: 'item'
});
if(objField.label === 'myLabel'){
    //Perform an action
}
if(objField.type === 'checkbox'){
    //Perform an action
}...
// Add additional code.
```

Field.getSelectOptions(options)

| Method Description | Returns an array of available options on a standard or custom select, multi-select, or radio field as key-value pairs. |

⚠️ **Important:** You can only use this method on a record in dynamic mode. For additional information on dynamic mode, see record.Record and SuiteScript 2.0 – Standard and Dynamic Modes.

<table>
<thead>
<tr>
<th>Returns</th>
<th>array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only the first 1,000 available options are returned in an array.</td>
<td></td>
</tr>
<tr>
<td>If there are more than 1,000 available options, an empty array [] is returned.</td>
<td></td>
</tr>
<tr>
<td>This function returns an array in the following format:</td>
<td></td>
</tr>
</tbody>
</table>
| ```javascript
| [{value: 5, text: 'abc'}, {value: 6, text: '123'}] |
| ```javascript |
| This function returns Type Error if the field is not a supported field for this method. |

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>None</th>
</tr>
</thead>
</table>

| Module | N/record Module |
Sibling Object Members | Field Object Members
---|---
Since | 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.filter</td>
<td>string</td>
<td>Required</td>
<td>The search string to filter the select options that are returned.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.operator</td>
<td>string</td>
<td>Required</td>
<td>The following operators are supported: &lt;br&gt;■ contains (default) &lt;br&gt;■ is &lt;br&gt;■ startswith</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** Filter values are case insensitive.

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

var options = objField.getSelectOptions({
    filter : 'C',
    operator : 'startswith'
});

//Perform an action with the options array
...
// Add additional code.
```

**Field.label**

| Property Description | Returns the UI label for a standard or custom field body or sublist field. |
Field.id

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the internal ID of a standard or custom body or sublist field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Field Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Field.type

**Property Description**  Returns the type of a body or sublist field.
For example, the value can return text, date, currency, select, checkbox, etc.
For more information on possible return values, see format.Type.
The maximum character limit for select field types is 801.

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
</table>

**Supported Script Types**  Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  N/record Module

**Sibling Object Members**  Field Object Members

**Since**  2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

var objField = objSublist.getField({
    fieldId: 'item'
});

// Perform an action with the objField.id value
...
// Add additional code.
```
Field.isMandatory

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns true if the standard or custom field is mandatory on the record form, or false otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean true</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Field Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

var objField = objSublist.getField({
    fieldId: 'item'
});

if(objField.isMandatory){
    var options = {
        title:'Incomplete Field:',
        message: 'Please complete this field.'
    }
    //Perform an action
    ...
    // Add additional code.
```
Field.sublistId

**Property Description**
Returns the sublist ID for the specified sublist field.

**Type**
string (read-only)

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/record Module

**Sibling Object Members**
Field Object Members

**Since**
2015.2

### Syntax

```
// Add additional code.
...
var objRecord = record.load({
  type: record.Type.SALES_ORDER,
  id: 275
});

var objSublist = objRecord.getSublist({
  sublistId: 'item'
});

var objField = objSublist.getField({
  fieldId: 'item'
});

//Perform an action with the objField.sublistId
...
// Add additional code.
```

Field.isDisplay

**Property Description**
Returns `true` if the field is visible on the record form, or `false` if it is not.

**Type**
boolean `true | false`

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.
record.Macro

**Object Description**
Encapsulates a record macro. For information about record macros, see the help topic [Overview of Record Action and Macro APIs](https://oracle.com).

Use the `Record.getMacro(options)` method to access the Macro object.

For a complete list of this object's methods and properties, see [Macro Object Members](#).

**Supported Script Types**
Client and server-side scripts.

For additional information, see the help topic [SuiteScript 2.0 Script Types](https://oracle.com).

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](https://oracle.com).

```javascript
// Add additional code
...
var myMacro = record.getMacro({id: 'calculateTax'})
...
// Add additional code
```

**Macro.execute(options)**

**Method Description**
Performs a macro operation and returns its result in a plain JavaScript object.

For information about record macros, see the help topic [Overview of Record Action and Macro APIs](https://oracle.com).

**Returns**
`{notifications: [], response: {}}`

**Supported Script Types**
Client and server scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](https://oracle.com).

**Module**
*N/record Module*

**Parent Object**
`record.Macro`

**Sibling Object Members**
Macro Object Members

**Since**
2018.2
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>The macro arguments.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code
...
timesheet.executeMacro({id:'copyFromWeek', params: {weekOf : '7/10/2017', copyExact : true}});
...
// Add additional code
```

Macro.execute.promise(options)

Method Description

Performs a macro operation asynchronously.

For information about record macros, see the help topic Overview of Record Action and Macro APIs.

Returns

Promise

Supported Script Types

Client-side scripts

For additional information, see the help topic SuiteScript 2.0 Client Script Type.

Module

N/record Module

Parent Object

record.Macro

Sibling Object Members

Macro Object Members

Since

2018.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>The macro arguments.</td>
</tr>
</tbody>
</table>
Macro(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Performs a macro operation and returns its result in a plain JavaScript object.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Substitute Macro with the name of the macro you are executing.</td>
</tr>
</tbody>
</table>
| Returns | (notifications: [], response: {})
| Supported Script Types | Client and server scripts
| For additional information, see the help topic [Overview of Record Action and Macro APIs](#).

| Module | N/record Module |
| Parent Object | record.Macro |
| Sibling Object Members | Macro Object Members |
| Since | 2018.2 |

**Parameters**

*Note:* The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>The macro arguments.</td>
</tr>
</tbody>
</table>

**Syntax**

*Important:* The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code
...
var calculateTax = recordObj.getMacro({id: 'calculateTax'});
calculateTax();
...
// Add additional code
```

Macro.promise(options)

| Method Description | Performs a macro operation asynchronously. |
**N/record Module**

**SuiteScript 2.0 API Reference**

**Note:** Substitute Macro with the name of the macro you are executing. [link to conceptual topic]

For information about record macros, see the help topic Overview of Record Action and Macro APIs.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Promise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client-side scripts</td>
</tr>
<tr>
<td></td>
<td>For additional information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/record Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Object</td>
<td>record.Macro</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Macro Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>The macro arguments.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code
...
var calculateTax = recordObj.getMacro({id: 'calculateTax'});
calculateTax.promise();
...
// Add additional code
```

**Macro.id**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The ID of the macro.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For information about record macros, see the help topic Overview of Record Action and Macro APIs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server scripts</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/record Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent Object</td>
<td>record.Macro</td>
</tr>
</tbody>
</table>
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code
...
var id = macro.id; // get the id of the macro
...
// Add additional code
```

### Macro.label

**Property Description**
The label of the macro.

For information about record macros, see the help topic Overview of Record Action and Macro APIs.

**Type**
string

**Supported Script Types**
Client and server scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/record Module

**Parent Object**
record.Macro

**Sibling Object Members**
Macro Object Members

**Since**
2018.2

---

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code
...
var label = macro.label; // get the label of the macro
...
// Add additional code
```

### Macro.description

**Property Description**
The description of the macro.

For information about record macros, see the help topic Overview of Record Action and Macro APIs.
**Type** string

**Supported Script Types** Client and server scripts
For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module** N/record Module

**Parent Object** record.Macro

**Sibling Object Members** Macro Object Members

**Since** 2018.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code
...
var description = macro.description; // get the description of the macro
...
// Add additional code
```

**Macro.attributes**

**Property Description** The defined attributes of the macro.
For information about record macros, see the help topic [Overview of Record Action and Macro APIs](#).

**Type** Object

**Supported Script Types** Client and server scripts
For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module** N/record Module

**Parent Object** record.Macro

**Sibling Object Members** Macro Object Members

**Since** 2018.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code
...
var attributes = macro.attributes; // get the attributes of the macro
...
```
record.Record

Load the record module when you want to work with NetSuite records.

Object Description

Encapsulates a NetSuite record.

There are two modes you can operate in when you create, copy, load, or transform a record with SuiteScript 2.0: standard mode and dynamic mode.

- When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in standard mode, the record's body fields and sublist line items are not sourced, calculated, and validated until the record is saved (submitted) with `Record.save(options)`.
  
When you work with a record in standard mode, you do not need to set values in any particular order. After submitting the record, NetSuite processes the record's body fields and sublist line items in the correct order, regardless of the organization of your script.

- When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in dynamic mode, the record's body fields and sublist line items are sourced, calculated, and validated in real-time. A record in dynamic mode emulates the behavior of a record in the UI.
  
When you work with a record in dynamic mode, it is important that you set values in the same order you would within the UI. If you fail to do this, your results may not be accurate.

The `record.create(options)`, `record.copy(options)`, `record.load(options)`, and `record.transform(options)` methods work in standard mode by default. If you want these methods to work in dynamic mode, you must pass in a specific argument. See the help topic for the applicable method for more information.

Use `record.Type` enum for multiple records. For help finding a record's internal ID, see the help topic How do I find a record's internal ID?

For more information about standard and dynamic modes, see the help topic SuiteScript 2.0 – Standard and Dynamic Modes

For a complete list of this object's methods and properties, see Record Object Members.

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/record Module

Since

2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.

... var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: '6',
    isDynamic: true
});
```
Record.cancelLine(options)

Method Description  Cancels the currently selected line on a sublist.
(dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic
Modes)

Returns  The record.Record object that called the method.

Supported Script Types  Client and server-side scripts

Governance  None

Module  N/record Module

Sibling Object Members  Record Object Members

Since  2015.2

Parameters

⚠️ Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.cancelLine({
  sublistId: 'item'
});
```
Record.commitLine(options)

Method Description
Commits the currently selected line on a sublist. (dynamic mode only — see the help topic SuiteScript 2.0 - Standard and Dynamic Modes)
When working in standard mode, set a sublist field using Record.setSublistValue(options).

Returns
The record.Record object that called the method.

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/record Module

Sibling Object Members
Record Object Members

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.commitLine({
    sublistId: 'item'
});
```
Record.executeMacro(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Performs macro operation and returns its result in a plain JavaScript object. For information about record macros, see the help topic Overview of Record Action and Macro APIs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>An object with the macro results or null.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server scripts For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Record Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The macro ID.</td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>The macro arguments.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code
...
if ('calculateTax' in macros) {
    macros.calculateTax();
}  
...
// Add additional code
```

Record.executeMacro.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Performs macro operation and returns its result in a plain JavaScript object. For information about record macros, see the help topic Overview of Record Action and Macro APIs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Promise</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The macro ID.</td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>The macro arguments.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code
...
if ('calculateTax' in macros) {
    macros.calculateTax.promise();
}
...
// Add additional code
```

**Record.findMatrixSublistLineWithValue(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the line number of the first instance where a specified value is found in a specified column of the matrix. Note that line and column indexing begins at 0 with SuiteScript 2.0. (dynamic and standard modes — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/record Module</td>
<td>Record Object Members</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

### Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The ID of the matrix field. See, How do I find a field's internal ID?</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>required</td>
<td>The value to search for.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number of the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var lineNumber = objRecord.findMatrixSublistLineWithValue({
  sublistId: 'item'
});
...
// Add additional code.
```

**Record.findSublistLineWithValue(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the line number for the first occurrence of a field value in a sublist. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>A line number as a number, or -1 if not found.</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>Date</td>
<td>string</td>
<td>array</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or not defined.</td>
</tr>
</tbody>
</table>

Syntax

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var lineNumber = objRecord.findSublistLineWithValue({
  sublistId: 'item',
  fieldId: 'item',
  value: 233
});
...
// Add additional code.
```

**Record.getCurrentMatrixSublistValue(options)**

**Method Description**

Gets the value for the currently selected line in the matrix.
Gets a numeric value for rate and ratehighprecision fields.

**Returns**
number | Date | string | array | boolean
true | false

**Supported Script Types**
Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the matrix field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

- **Error Code**: SSS_MISSING_REQD_ARGUMENT
  - **Thrown If**: A required argument is missing or undefined.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var matrixValue = objRecord.getCurrentMatrixSublistValue({
  sublistId: 'item',
  fieldId: 'item',
  column: 12
});
...
```
// Add additional code.

Record.getCurrentSublistField(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns metadata about a sublist field.</th>
<th>(dynamic mode only— see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>record.Field</td>
<td>________________________________________________________________----------------------------------------------------------------</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
<td>________________________________________________________________----------------------------------------------------------------</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
<td>________________________________________________________________----------------------------------------------------------------</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
<td>________________________________________________________________----------------------------------------------------------------</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Record Object Members</td>
<td>________________________________________________________________----------------------------------------------------------------</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
<td>________________________________________________________________----------------------------------------------------------------</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See, How do I find a field's internal ID?</td>
<td></td>
</tr>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var sublistFieldMetadata = objRecord.getCurrentSublistField({
    sublistId: 'item',
    fieldId: 'item',
});
```
Record.getCurrentSublistIndex(options)

**Method Description**
Returns the line number of the currently selected line. Note that line indexing begins at 0 with SuiteScript 2.0.  
(dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**
number

**Supported Script Types**
Client and server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

**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>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

```
// Add additional code.
...
var currIndex = objRecord.getCurrentSublistIndex({
    sublistId: 'item'
});
...
// Add additional code.
```
Record.getCurrentSublistSubrecord(options)

**Method Description**
Gets the subrecord for the associated sublist field on the current line.
(dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**
record.Record

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

**Parameters**

- **options.sublistId**
  - **Type:** string
  - **Required / Optional:** required
  - **Description:** The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.

- **options.fieldId**
  - **Type:** string
  - **Required / Optional:** required
  - **Description:** The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?

**Syntax**

```javascript
// Add additional code.
... var objSubrecord = objRecord.getCurrentSublistSubrecord({
  sublistId: 'item',
  fieldId: 'item'
});
... // Add additional code.
```

Record.getCurrentSublistText(options)

**Method Description**
Returns a text representation of the field value in the currently selected line.
(dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

Gets a string value with a "%" for rate and ratehighprecision fields.

<table>
<thead>
<tr>
<th>Returns</th>
<th>string</th>
</tr>
</thead>
</table>

**Note:** For multiselect fields, returns an array.

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/record Module

**Sibling Object Members**

Record Object Members

**Since**

2015.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var fieldName = objRecord.getCurrentSublistText({
    sublistId: 'item',
    fieldId: 'item'
});
```
Record.getCurrentSublistValue(options)

Method Description
Returns the value of a sublist field on the currently selected sublist line. (dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

Returns
number | Date | string | array | boolean true | false

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/record Module

Sibling Object Members
Record Object Members

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field’s internal ID?</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

// Add additional code.
Record.getField(options)

**Method Description**

Returns a field object from a record.
(dynamic and standard modes — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**

record.Field

**Supported Script Types**

Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/record Module

**Sibling Object Members**

Record Object Members

**Since**

2015.2

**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>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objField = objRecord.getField({
  options: {fieldId: 'item'}
});
```
### Record.getFields()

**Method Description**

Returns the body field names (internal ids) of all the fields in the record, including machine header field and matrix header fields.

(dynamic and standard modes — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**

string[]

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/record Module

**Sibling Object Members**

Record Object Members

**Since**

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objFields = objRecord.getFields();
...
// Add additional code.
```

### Record.getLineCount(options)

**Method Description**

Returns the number of lines in a sublist.

(standard and dynamic mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**

number

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/record Module

**Sibling Object Members**

Record Object Members
Since 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The macro ID.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var numLines = objRecord.getLineCount({
    sublistId: 'item'
});
...
// Add additional code.
```

Record.getMacro(options)

Method Description

Provides a macro to be executed.

For information about record macros, see the help topic Overview of Record Action and Macro APIs.

Returns

Function to be executed for the macro.

Supported Script Types

Client and server scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

Module

N/record Module

Sibling Object Members

Record Object Members

Since 2018.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The macro ID.</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_MACRO_ID</td>
<td>A macro does not exist on the record.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code
...
var macro = recordObj.getMacro({id: 'calculateTax'});
...
// Add additional code
```

Record.getMacros(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Provides a plain JavaScript object of available macro objects defined for a record type, indexed by the Macro ID. The object returns one or more record.Macro objects. If there are no macros available for the specified record type, an empty object is returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server scripts For additional information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>
Record.getMatrixHeaderCount(options)

**Method Description**
Returns the number of columns for the specified matrix.
(standard and dynamic mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**
number

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

**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>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

```
// Add additional code.
```

---

... var macroList = recordObj.getMacros(); ...

// Add additional code
var numLines = objRecord.getMatrixHeaderCount(
    sublistId: 'item',
    fieldId: 'item'
));
...
// Add additional code.

Record.getMatrixHeaderField(options)

**Method Description**
Gets the field for the specified header in the matrix.
(standard and dynamic mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**
record.Field

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field.</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objField = objRecord.getMatrixHeaderField(
    sublistId: 'item',
    fieldId: 'item',
    column: 12
);
... // Add additional code.
```

### Record.getMatrixHeaderValue(options)

**Method Description**

Gets the value for the associated header in the matrix.

(standard and dynamic mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

Gets a numeric value for rate and ratehighprecision fields.

**Returns**

number | Date | string | array | boolean

true | false

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/record Module

**Sibling Object Members**

Record Object Members

**Since**

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See, How do I find a field's internal ID?</td>
<td></td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
var value = objRecord.getMatrixHeaderValue({
  sublistId: 'item',
  fieldId: 'item',
  column: 12
});
...
// Add additional code.
```

**Record.getMatrixSublistField(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the field for the specified sublist in the matrix.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(standard and dynamic mode — see the help topic <a href="#">SuiteScript 2.0 - Standard and Dynamic Modes</a>)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>record.Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>None</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Record Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Using the SuiteScript Records Browser</a>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See, How do I find a field's internal ID?</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objField = objRecord.getMatrixSublistField({
  sublistId: 'item',
  fieldId: 'item',
  column: 12,
  line: 3
});
...
// Add additional code.
```

**Record.getMatrixSublistValue(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the value for the associated field in the matrix. (standard and dynamic mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Record Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var value = objRecord.getMatrixSublistValue({
  sublistId: 'item',
  fieldId: 'item',
  column: 12,
  line: 3
});
...
// Add additional code.
```

Record.getSublist(options)

Method Description

Returns the specified sublist.

(standard and dynamic mode — see the help topic SuiteScript 2.0 - Standard and Dynamic Modes)
**Returns**  record.Sublist

**Supported Script Types**  Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**  None

**Module**  N/record Module

**Sibling Object Members**  Record Object Members

**Since**  2015.2

## Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Using the SuiteScript Records Browser</a>.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
var objSublist = objRecord.getSublist({
  sublistId: 'item'
});
...
// Add additional code.
```

### Record.getSublists()

**Method Description**  Returns all the names of all the sublists.

(standard and dynamic mode — see the help topic [SuiteScript 2.0 – Standard and Dynamic Modes](#))

<table>
<thead>
<tr>
<th>Returns</th>
<th>string[]</th>
</tr>
</thead>
</table>

**Supported Script Types**  Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**  None

**Module**  N/record Module
Record.getSublistField(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a field object from a sublist. (standard and dynamic mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</td>
<td>record.Field</td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/record Module</td>
<td>Record Object Members</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objField = objRecord.getSublistField({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
// Add additional code.
```

Record.getSublistFields(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns all the field names in a sublist.</td>
<td>string[]</td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/record Module</td>
<td>Record Object Members</td>
<td>2015.2</td>
</tr>
<tr>
<td>(standard and dynamic mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</td>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Parameters

ℹ️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples. For other samples, see Sublist.getColumn(options).

```javascript
// Add additional code.
...
var field = objRecord.getSublistFields({
   sublistId: 'item'
});
...
// Add additional code.
```

Record.getSublistSubrecord(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the subrecord associated with a sublist field. (standard mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When working in dynamic mode, get a sublist subrecord using the following methods:</td>
</tr>
<tr>
<td></td>
<td>1. Record.selectLine(options)</td>
</tr>
<tr>
<td></td>
<td>2. Record.hasCurrentSublistSubrecord(options)</td>
</tr>
<tr>
<td></td>
<td>3. Record.getCurrentSublistSubrecord(options)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>record.Record</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/record Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Sibling Object Members</th>
<th>Record Object Members</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2015.2</th>
</tr>
</thead>
</table>

Parameters

ℹ️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2015.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>-------------</td>
<td>----------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objSubRecord = objRecord.getSublistSubrecord({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
// Add additional code.
```

**Record.getSublistText(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the value of a sublist field in a text representation. (standard mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</td>
<td>string</td>
</tr>
<tr>
<td>Gets a string value with a &quot;%&quot; for rate and ratehighprecision fields.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For multiselect fields, returns an array.

**Supported Script Types**

- Client and server-side scripts.
  - For more information, see the help topic SuiteScript 2.0 Script Types.
  - Limitations exist on how this method can be used in standard (deferredDynamic) mode. For details, refer to the description of the SSS_INVALID_API_USAGE error code in the Errors table.
  - In dynamic mode, you can use getSublistText() without limitation.

**Governance**

- None

**Module**

- N/record Module
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_API_USAGE</td>
<td>Invoked in certain cases when deferredDynamic mode is being used.</td>
</tr>
<tr>
<td></td>
<td>For example, if Record.isDynamic is set to false, this error can be invoked in both of the following situations:</td>
</tr>
<tr>
<td></td>
<td>- If the record object was created by record.copy(), record.create(), or record.transform(), and the script attempts to use getSublistText() without first using setSublistText() for the same field.</td>
</tr>
<tr>
<td></td>
<td>- If the record object was created by record.load(), and the script uses setSublistValue() on a field before using getSublistText() for the same field.</td>
</tr>
<tr>
<td></td>
<td>This guidance also affects user event scripts that instantiate records by using the newRecord or oldRecord object provided by the script context. These records always use deferredDynamic mode. For that reason, this error appears in both of the following situations:</td>
</tr>
<tr>
<td></td>
<td>- When a user event script executes on a record that is being newly created, and the script attempts to use getSublistText() without first using setSublistText() for the same field.</td>
</tr>
<tr>
<td></td>
<td>- When a user event script executes on an existing record, and the script uses setSublistValue() on a field before using getSublistText() for the same field.</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 – Standard and Dynamic Modes.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see *N/record Module Script Samples.*

```javascript
// Add additional code.
...
var sublistFieldName = objRecord.getSublistText({
  sublistId: 'item',
  fieldId: 'item',
  line: 3
});
...
// Add additional code.
```

**Record.getSublistValue(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the value of a sublist field.</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/record Module</td>
<td>Record Object Members</td>
<td>2015.2</td>
</tr>
<tr>
<td>(dynamic and standard modes — see the help topic <em>SuiteScript 2.0 – Standard and Dynamic Modes</em>)</td>
<td>Date</td>
<td>For more information, see the help topic <em>SuiteScript 2.0 Script Types.</em></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gets a numeric value for rate and ratehighprecision fields.</td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>boolean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>false</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic <em>Using the SuiteScript Records Browser.</em></td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field’s internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_API_USAGE</td>
<td>Invoked prior to using setSublistValue in standard record mode.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
var sublistFieldValue = objRecord.getSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    line: 3
});
...
// Add additional code.
```

**Record.getSubrecord(options)**

Method Description

- Gets the subrecord for the associated field.
- This method is not available for subrecords.
- (dynamic and standard mode — see the help topic [SuiteScript 2.0 – Standard and Dynamic Modes](#))

Returns

- record.Record

Supported Script Types

- Client and server-side scripts
- For more information, see the help topic [SuiteScript 2.0 Script Types](#).

Governance

- None

Module

- N/record Module

Sibling Object Members

- Record Object Members

Since

- 2015.2

Parameters

⚠️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>FIELD_1_IS_NOT_A_SUBRECORD_FIELD</td>
<td>The specified field is not a subrecord field.</td>
</tr>
<tr>
<td>FIELD_1_IS_DISABLED_YOU_CANNOT_APPLY_SUBRECORD_OPERATION_ON_THIS_FIELD</td>
<td>The specified field is disabled.</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var sublistFieldValue = objRecord.getSubrecord({
  fieldId: 'idnumber'
});
...
// Add additional code.
```

### Record.getText(options)

**Method Description**

Returns the text representation of a field value.

- (dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)
- Gets a string value with a "%" for rate and ratehighprecision fields.

**Returns**

- string

**Note:** For multiselect fields, returns an array.

**Supported Script Types**

Client and server-side scripts.

For more information, see the help topic SuiteScript 2.0 Script Types.

In dynamic mode, you can use `getText()` without limitation but, in standard mode, limitations exist. In standard mode, you can use this method only in the following cases:

- You can use `getText()` on any field where the script has already used `setText()`.
- If you are loading or copying a record, you can use `getText()` on any field except those where the script has already changed the value by using `setValue()`.

For more details, refer to the description of the SSS_INVALID_API_USAGE error code in the Errors table.
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See, <a href="#">How do I find a field's internal ID?</a></td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_API_USAGE</td>
<td>Invoked in certain cases when standard mode is being used.</td>
</tr>
<tr>
<td></td>
<td>For example, if <code>Record.isDynamic</code> is set to false, the SSS_INVALID_API_USAGE error can be invoked in the following situations:</td>
</tr>
<tr>
<td></td>
<td>■ If the record object was created by <code>record.create()</code> or <code>record.transform()</code>, and the script attempts to use <code>getText()</code> without first using <code>setText()</code> for the same field.</td>
</tr>
<tr>
<td></td>
<td>■ The record object was created by <code>record.copy()</code> or <code>record.load()</code>, and the script uses <code>setValue()</code> on a field before using <code>getText()</code> for the same field.</td>
</tr>
<tr>
<td></td>
<td>Similar guidance affects user event scripts that instantiate records by using the <code>newRecord</code> or <code>oldRecord</code> object provided by the script context. In these cases, standard mode is always used. For that reason, the SSS_INVALID_API_USAGE error appears when a user event executes on one of these objects in the following situations:</td>
</tr>
<tr>
<td></td>
<td>■ When the script executes on a record that is being created, and the script attempts to use <code>getText()</code> without first using <code>setText()</code> for the same field.</td>
</tr>
<tr>
<td></td>
<td>■ When the script executes on an existing record or on a record being created through copying, and the script uses <code>setValue()</code> on a field before using <code>getText()</code> for the same field.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Syntax Sample 1
// Add additional code.
...```
```javascript
var fieldIdname = objRecord.getText({
    fieldId: 'item'
});
...
// Add additional code.

// Syntax Sample 2
// Add additional code.
...
myString = 'Date is: ' + record.getText({fieldId: 'datechanged'});
// "Date is: 3/27/2017 9:55:38am"
...
// Add additional code
```

---

### Record.getValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the value of a field.</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/record Module</td>
<td>Record Object Members</td>
<td>2015.2</td>
</tr>
<tr>
<td>Gets a numeric value for rate and ratehighprecision fields.</td>
<td>Date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>array</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>boolean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>true</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Returns a JavaScript Date object for date/time field queries.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>To return a string for date/time field queries, use Record.getText(options). Date/time fields: DATE, DATETIME, DATETIMETZ, TIMEOFDAY.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Note: If the returned date object is implicitly converted to a string, the value is converted using the browser’s setting for time zone.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checkbox fields return values of T or F. If you include checkbox field return values in scripts, be sure to use T and F, instead of boolean values, true and false.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See, How do I find a field's internal ID?</td>
<td></td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_API_USAGE</td>
<td>Invoked in standard mode, if you use setText on a field and then use getValue on the same field.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var value = objRecord.getValue(
   fieldId: 'item'
);
...
// Add additional code.
```

Record.hasCurrentSublistSubrecord(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a value indicating whether the associated sublist field has a subrecord on the current line. (dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</td>
<td>boolean <code>true</code></td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/record Module</td>
<td>Record Object Members</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

⚠️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.
### Record.hasSublistSubrecord(options)

**Method Description**
Returns a value indicating whether the associated sublist field contains a subrecord.

This method is not available for subrecords.

(standard mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Returns**
boolean `true` | `false`

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Record.hasSubrecord(options)

**Method Description**

Returns a value indicating whether the field contains a subrecord.

This method is not available for subrecords.

*(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)*

**Returns**

`boolean` `true` | `false`

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

`N/record Module`

**Sibling Object Members**

`Record Object Members`

**Since**

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.fieldId</code></td>
<td><code>string</code></td>
<td>required</td>
<td>The internal ID of the field that may contain a subrecord.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Record.insertLine(options)

Method Description
Inserts a sublist line.
When you insert a line with this method, all succeeding lines are moved and the total line count is increased. Essentially, succeeding lines are committed to a new sublist line with a new line number.
(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

Returns
record.Record

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/record Module

Sibling Object Members
Record Object Members

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Parameter Types

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number to insert. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.#ignoreRecalc</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, scripting recalculation is ignored. The default value is false.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example that uses insertLine(), see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
objRecord.insertLine({
  sublistId: 'attendee',
  line: 2,
});
objRecord.setCurrentSublistValue({
  sublistId: 'attendee',
  fieldId: 'attendee',
  value: 838
});
objRecord.commitLine({
  sublistId: 'attendee'
});
...
// Add additional code.
```

For script examples that use other N/record methods, see [N/record Module Script Samples](#).

### Record.removeCurrentSublistSubrecord(options)

**Method Description**

Removes the subrecord for the associated sublist field on the current line. This method is not available for subrecords.

(dynamic mode only — see the help topic [SuiteScript 2.0 – Standard and Dynamic Modes](#))

**Returns**

(record.Record)
Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/record Module

Sibling Object Members
Record Object Members

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field’s internal ID?</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.removeCurrentSublistSubrecord{
    sublistId: 'item',
    fieldId: 'item'
};
...
// Add additional code.

Record.removeLine(options)

Method Description
Removes a sublist line.
(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

Returns
record.Record

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/record Module

Sibling Object Members
Record Object Members

Since
2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number of the sublist to remove. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.ignoreRecalc</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, scripting recalculation is ignored. The default value is false.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.removeLine({
    sublistId: 'item',
    line: 3,
    ignoreRecalc: true
});
...
// Add additional code.
```
Record.removeSublistSubrecord(options)

Method Description
Removes the subrecord for the associated sublist field. (standard mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

When working in dynamic mode, remove a sublist subrecord using the following methods:

1. Record.selectLine(options)
2. Record.hasCurrentSublistSubrecord(options)
3. Record.removeCurrentSublistSubrecord(options)
4. Record.commitLine(options)

Returns
record.Record

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/record Module

Sibling Object Members
Record Object Members

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number in the sublist that contains the subrecord to remove. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
```
Record.removeSubrecord(options)

**Method Description**

Removes the subrecord for the associated field.

This method is not available for subrecords.

*(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)*

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>record.Record</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>

For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Governance</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/record Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sibling Object Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Object Members</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.removeSublistSubrecord({
    sublistId: 'item',
    fieldid: 'item',
    line: 3
});
...```

---

SuiteScript 2.0 API Reference
Record.save(options)

Method Description
Submits a new record or saves edits to an existing record.
When working with records in standard mode, you must submit and then load the
record to obtain sourced, validated, and calculated field values.
This method is not available to subrecords.

**Note:** This method has an asynchronous counterpart you can use with client
scripts. See Record.save.promise(options).

Returns
A number representing the internal ID of the new or updated record.

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
Transaction records: 20 usage units
Custom records: 4 usage units
All other records: 10 usage units

Module
N/record Module

Sibling Object Members
Record Object Members

Since
2015.2

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>options.enableSourcing</td>
<td>boolean</td>
<td>optional</td>
<td>Enables sourcing during the record update.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td></td>
<td>If set to true, sources dependent field information for empty fields.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td>Defaults to false – dependent field values are not sourced.</td>
<td></td>
</tr>
<tr>
<td>options.ignoreMandatoryFields</td>
<td>boolean</td>
<td>optional</td>
<td>Disables mandatory field validation for this save operation.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>If set to <code>true</code>, all standard and custom fields that were made mandatory through customization are ignored. All fields that were made mandatory through company preferences are also ignored. By default, this parameter is <code>false</code>.</td>
</tr>
</tbody>
</table>

**Important:** Use the `ignoreMandatoryFields` argument with caution. This argument should be used mostly with Scheduled scripts, rather than User Event scripts. This ensures that UI users do not bypass the business logic enforced through form customization.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
var recordId = objRecord.save({
  enableSourcing: true,
  ignoreMandatoryFields: true
});
...
// Add additional code.
```

### Record.save.promise(options)

**Method Description**

Submits a new record asynchronously or saves edits to an existing record asynchronously.

This method is not available to subrecords.

**Note:** For information about the parameters and errors thrown for this method, see [Record.save(options)](#). For more information on promises, see [Promise Object](#).

<table>
<thead>
<tr>
<th>Returns</th>
<th>Promise</th>
</tr>
</thead>
</table>

**Supported Script Types**

Client-side scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

**Governance**

Transaction records: 20 usage units

Custom records: 4 usage units

All other records: 10 usage units

**Module**

N/record Module

**Sibling Object Members**

Record Object Members
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.enableSourcing</td>
<td>boolean</td>
<td>optional</td>
<td>Enables sourcing during the record update. If set to <code>true</code>, sources dependent field information for empty fields. Defaults to <code>false</code> – dependent field values are not sourced.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.ignoreMandatoryFields</td>
<td>boolean</td>
<td>optional</td>
<td>Disables mandatory field validation for this save operation. If set to <code>true</code>, all standard and custom fields that were made mandatory through customization are ignored. By default, this parameter is <code>false</code>.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Important:** This parameter applies to records in standard mode only. When working with records in dynamic mode, field values are always sourced and the value you provide for `enableSourcing` is ignored. See the help topic SuiteScript 2.0 – Standard and Dynamic Modes.

**Important:** Use the `ignoreMandatoryFields` argument with caution. This argument should be used mostly with Scheduled scripts, rather than User Event scripts. This ensures that UI users do not bypass the business logic enforced through form customization.

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```javascript
// Add additional code.
...
var recordId = objRecord.save.promise({
    enableSourcing: true,
    ignoreMandatoryFields: true
});
...
// Add additional code.
```
Record.selectLine(options)

### Method Description
Selects an existing line in a sublist. (dynamic mode only — see the help topic [SuiteScript 2.0 – Standard and Dynamic Modes](#)).

When working in standard mode, set a sublist field using `Record.setSublistValue(options)`.

### Returns
`record.Record`

### Supported Script Types
Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

### Governance
None

### Module
`N/record Module`

### Sibling Object Members
`Record Object Members`

### Since
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic <a href="#">Using the SuiteScript Records Browser</a>.</td>
<td></td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number to select in the sublist. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
var lineNum = objRecord.selectLine({
```
Record.selectNewLine(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Selects a new line at the end of a sublist. (dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>record.Record</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Record Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

![Note: The options parameter is a JavaScript object.]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

Syntax

![Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.]

// Add additional code.
... var lineNum = objRecord.selectNewLine({
    sublistId: 'item'
});
...
// Add additional code.

Record.setCurrentMatrixSublistValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Sets the value for the line currently selected in the matrix. (dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes) Sets a string value with a &quot;%&quot; for rate and ratehighprecision fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>record.Record</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Record Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>Date</td>
<td>string</td>
<td>array</td>
</tr>
</tbody>
</table>
### 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></td>
<td></td>
<td></td>
<td>- Checkbox fields accept Boolean values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Date and DateTime fields accept Date values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- Integer, Float, Currency and Percent fields accept number values.</td>
<td></td>
</tr>
</tbody>
</table>

**options.ignore**
- **Type**: `boolean`
- **Required / Optional**: `optional`
- **Description**: If set to `true`, the field change and slaving event is ignored. By default, this value is `false`.
- **Since**: 2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

**Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
objRecord.setCurrentMatrixSublistValue(
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
);
...
// Add additional code.
```

### Record.setCurrentSublistText(options)

**Method Description**: Sets the value for the field in the currently selected line by a text representation. (dynamic mode only — see the help topic [SuiteScript 2.0 – Standard and Dynamic Modes](#))

- Sets a string value with a "%" for rate and ratehighprecision fields.

**Returns**: `record.Record`

**Supported Script Types**: Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.text</td>
<td>string</td>
<td>required</td>
<td>The text to set the value to.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.ignoreFieldChange</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, the field change and slaving event is ignored. By default, this value is false.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>A_SCRIPT_IS_ATTEMPTING_TO_EDIT_THE_SUBLIST_THIS_SUBLIST_IS_CURRENTLY_IN_READ_ONLY</td>
<td>Only sublist field.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.setCurrentSublistText(
    sublistId: 'item',
    fieldId: 'item',
    text: 'value',
    ignoreFieldChange: true
);```
Record.setCurrentSublistValue(options)

Sets the value for the field in the currently selected line. (dynamic mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

When working in standard mode, set a sublist field using Record.setSublistValue(options).

Important: When you edit a sublist line with SuiteScript, it triggers an internal validation of the sublist line. If the line validation fails, the script also fails. For example, if your script edits a closed catch up period, the validation fails and prevents SuiteScript from editing the closed catch up period.

Sets a numeric value for rate and ratehighprecision fields.

Returns record.Record

Supported Script Types Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance None

Module N/record Module

Sibling Object Members Record Object Members

Since 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>Date</td>
<td>string</td>
<td>array</td>
</tr>
</tbody>
</table>
### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.ignoreFieldChange</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, the field change and slaving event is ignored. By default, this value is false.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.setCurrentSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true
});
...
// Add additional code.
```

### Record.setMatrixHeaderValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the value for the associated header in the matrix. (dynamic and standard modes — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)</td>
<td>record.Record</td>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See, How do I find a field's internal ID?</td>
<td></td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td></td>
</tr>
<tr>
<td>options.value</td>
<td>number</td>
<td>required</td>
<td>The value to set the field to.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>Date</td>
<td></td>
<td>The value type must correspond to the field type being set. For example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
<td>■ Text, Radio and Select fields accept string values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>array</td>
<td></td>
<td>■ Checkbox fields accept Boolean values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>boolean</td>
<td></td>
<td>■ Date and DateTime fields accept Date values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>true</td>
<td></td>
<td>■ Integer, Float, Currency and Percent fields accept number values.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td>Errors for field changes can be ignored, if this option is set to true.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is false.</td>
<td></td>
</tr>
<tr>
<td>options.@ignoreFieldChange</td>
<td>boolean</td>
<td>optional</td>
<td>If set to true, the field change and slaving event is ignored.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td></td>
<td>Errors for field changes can be ignored, if this option is set to true.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td>By default, this value is false.</td>
<td></td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.setMatrixHeaderValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 3,
    value: false,
    ignoreFieldChange: true,
    forceSyncSourcing: true
});
...
// Add additional code.
```

Record.setMatrixSublistValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sets the value for the associated field in the matrix. (standard mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes) Sets a numeric value for rate and ratehighprecision fields.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>record.Record</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/record Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sibling Object Members</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record Object Members</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

⚠️ Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublistId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist that contains the matrix.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td></td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the matrix field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See, How do I find a field's internal ID?</td>
<td></td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------------------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>options.column</td>
<td>number</td>
<td>required</td>
<td>The column number for the field. Note that column indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
| options.value   | number | Date | string | array | boolean | true | false | required | The value to set the field to. The value type must correspond to the field type being set. For example:
- Text, Radio and Select fields accept string values.
- Checkbox fields accept Boolean values.
- Date and DateTime fields accept Date values.
- Integer, Float, Currency and Percent fields accept number values. | 2015.2 |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

```
// Add additional code.
...
objRecord.setMatrixSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    column: 12,
    line: 3,
    value: true
});
...
// Add additional code.
```

Record.setSublistText(options)

**Method Description**

Sets the value of a sublist field by a text representation. (standard mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

When working in dynamic mode, set a sublist field text using the following methods:
1. `Record.selectLine(options)`
2. `Record.setCurrentSublistText(options)`
3. `Record.commitLine(options)`

Sets a string value with a "%" for rate and ratehighprecision fields.

**Returns**

`record.Record`

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/record Module

**Sibling Object Members**

Record Object Members

**Since**

2015.2

### Parameters

**Note:** The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.sublistId</code></td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist. This value is displayed in the Records Browser. For more information, see the help topic Using the SuiteScript Records Browser.</td>
<td>2015.2</td>
</tr>
<tr>
<td><code>options.fieldId</code></td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom sublist field. See, How do I find a field's internal ID?</td>
<td>2015.2</td>
</tr>
<tr>
<td><code>options.line</code></td>
<td>number</td>
<td>required</td>
<td>The line number for the field. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
<tr>
<td><code>options.text</code></td>
<td>string</td>
<td>required</td>
<td>The text to set the value to.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>SSS_MISSING_REQD_ARGUMENT</code></td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td><code>SSS_INVALID_SUBLIST_OPERATION</code></td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
```
Record.setSublistValue(options)

**Method Description**
Sets the value of a sublist field. (standard mode only — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

When working in dynamic mode, set a sublist field value using the following methods:

1. Record.selectLine(options)
2. Record.setCurrentSublistValue(options)
3. Record.commitLine(options)

**Important:** When you edit a sublist line with SuiteScript, it triggers an internal validation of the sublist line. If the line validation fails, the script also fails. For example, if your script edits a closed catch up period, the validation fails and prevents SuiteScript from editing the closed catch up period.

Sets a numeric value for rate and ratehighprecision fields.

**Returns**
record.Record

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

**Parameters**

<table>
<thead>
<tr>
<th>i</th>
<th>Note: The options parameter is a JavaScript object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parameter</td>
<td>Type</td>
</tr>
<tr>
<td>options.sublistId</td>
<td>string</td>
</tr>
<tr>
<td>options.fieldId</td>
<td>string</td>
</tr>
</tbody>
</table>
### 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>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number of the sublist. Note that line indexing begins at 0 with SuiteScript 2.0.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
| options.value | number | Date | string | array | boolean true | false | required | The value to set the sublist field to. The value type must correspond to the field type being set. For example:  
  - Text, Radio and Select fields accept string values.  
  - Checkbox fields accept Boolean values.  
  - Date and DateTime fields accept Date values.  
  - Integer, Float, Currency and Percent fields accept number values. | 2015.2|

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The <code>options.value</code> type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
<tr>
<td>SSS_INVALID_SUBLIST_OPERATION</td>
<td>A required argument is invalid or the sublist is not editable.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
objRecord.setSublistValue({
    sublistId: 'item',
    fieldId: 'item',
    line: 3,
    value: true
});
...
// Add additional code.
```

### Record.setText(options)

**Method Description:**

Sets the value of the field by a text representation.

*(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)*

Sets a string value with a "%" for rate and ratehighprecision fields.
Returns | record.Record
---|---
Supported Script Types | Client and server-side scripts
For more information, see the help topic [SuiteScript 2.0 Script Types](#).
Governance | None
Module | N/record Module
Sibling Object Members | Record Object Members
Since | 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field. See, <a href="#">How do I find a field’s internal ID?</a></td>
</tr>
<tr>
<td>options.text</td>
<td>string</td>
<td>required</td>
<td>The text or texts to change the field value to. If the field type is <strong>multiselect</strong>: □ This parameter accepts an array of string values. □ This parameter accepts a null value. Passing in null deselects all currently selected values. If the field type is <strong>not multiselect</strong>, this parameter accepts only a single string value.</td>
</tr>
<tr>
<td>options.<strong>ignore</strong>FieldChange</td>
<td>boolean</td>
<td>optional</td>
<td>If set to <strong>true</strong>, the field change and slaving event is ignored. By default, this value is <strong>false</strong>.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
```
objRecord.setText({
    fieldId: 'item',
    text: 'value',
    ignoreFieldChange: true
});
...
// Add additional code.

Record.setValue(options)

### Method Description
Sets the value of a field.

(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

<table>
<thead>
<tr>
<th>Returns</th>
<th>record.Record</th>
</tr>
</thead>
</table>

| Supported Script Types | Client and server-side scripts |

For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Governance</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/record Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Sibling Object Members</th>
<th>Record Object Members</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2015.2</th>
</tr>
</thead>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a standard or custom body field.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See, How do I find a field's internal ID?</td>
<td></td>
</tr>
</tbody>
</table>
| options.value   | number | Date | string | array | boolean | true | false | required   | The value to set the field to. The value type must correspond to the field type being set. For example:

- Text, Radio, Select and Multi-Select fields accept string values.
- Checkbox fields accept Boolean values.
- Date and DateTime fields accept Date values.
- Integer, Float, Currency and Percent fields accept number values. | 2015.2|
### 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>options.@ignoreFieldChange</td>
<td>boolean</td>
<td>optional</td>
<td>If set to <code>true</code>, the field change and slaving event is ignored. By default, this value is <code>false</code>.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_FLD_VALUE</td>
<td>The options.value type does not match the field type.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
// Add additional code.
...
objRecord.setValue({
    fieldId: 'item',
    value: true,
    ignoreFieldChange: true
});
...
// Add additional code.
```

### Record.id

**Property Description**

The internal ID of a specific record.

This property is not available to subrecords.

(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Type**

number (read-only)

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/record Module

**Sibling Object Members**

Record Object Members

**Since**

2015.2

### Record.isDynamic

**Property Description**

Indicates whether the record is in dynamic or standard mode.
If set to `true`, the record is currently in dynamic mode. If set to `false`, the record is currently in standard mode.

- When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in standard mode, the record's body fields and sublist line items are not sourced, calculated, and validated until the record is saved (submitted) with `Record.save(options)`.

  When you work with a record in standard mode, you do not need to set values in any particular order. After submitting the record, NetSuite processes the record's body fields and sublist line items in the correct order, regardless of the organization of your script.

- When a SuiteScript 2.0 script creates, copies, loads, or transforms a record in dynamic mode, the record's body fields and sublist line items are sourced, calculated, and validated in real-time. A record in dynamic mode emulates the behavior of a record in the UI.

  When you work with a record in dynamic mode, it is important that you set values in the same order you would within the UI. If you fail to do this, your results may not be accurate.

This value is set when the record is created or accessed.

(dynamic and standard mode — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean <code>true</code></th>
<th><code>false</code> (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
<td></td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>Record Object Members</td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
if (record.isDynamic) {
    ...
}
...```

**Record.type**

**Property Description**

The record type. Note the following:

- When working with an instance of a standard NetSuite record type, set this value by using the `record.Type` enum.

- When working with an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.
This property is not available to subrecords.
(dynamic and standard modes — see the help topic SuiteScript 2.0 – Standard and Dynamic Modes)

**Type**
string (read-only)

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/record Module

**Sibling Object Members**
Record Object Members

**Since**
2015.2

## Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...

// Start the process of creating an employee record.

var employeeRecord = record.create({
    type: record.Type.EMPLOYEE,
    isDynamic: true
});

// Start the process of creating an instance of a custom record type.

var customRecord = record.create({
    type: 'customrecord_book',
    isDynamic: true
});

...
// Add additional code.
```

**Note:** Supported standard record types are described in the SuiteScript Records Browser. Refer also to SuiteScript Supported Records. For help working with custom record types, see the help topic Custom Record.

### record.Sublist

**Object Description**
Encapsulates a sublist on a standard or custom record.
For a complete list of this object's methods and properties, see Sublist Object Members.

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.
Module: N/record Module

Since: 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
  type: record.Type.SALES_ORDER,
  id: 275
});

var objSublist = objRecord.getSublist({
  sublistId: 'item'
});

if(objSublist.type === 'inlineeditor'){
  //Perform an action
}
...
// Add additional code.
```

**Sublist.getColumn(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Sibling Object Members</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a column in the sublist.</td>
<td>record.Column</td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/record Module</td>
<td>Sublist Object Members</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fieldId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the column field in the sublist. Must be a valid field ID</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see *N/record Module Script Samples.*

#### Example 1

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

var objColumn = objSublist.getColumn({
    fieldId: 'item'
});

if(objColumn.type === 'checkbox'){
    //Perform an action
}
...
// Add additional code.
```

#### Example 2

This example loops through each line of the items sublist on a sales order record.

```javascript
// Add additional code
...
onRequest: function(context) {
    var recordObj = record.create({type: record.Type.SALES_ORDER});
    var columnList = recordObj.getSublistFields({sublistId: 'item'});
    var sublistObj = recordObj.getSublist({sublistId: 'item'});

    for (var i = 0; i < columnList.length; i++) {
        var columnId = columnList[i];
        var columnObj = sublistObj.getColumn({fieldId: columnId});
        if (columnObj !== null) {
            log.debug(('[Column id] = ' + columnObj.id + ' [Column type] = ' + columnObj.type + ' [Column label] = ' + columnObj.label);
        }
    }
    ...
// Add additional code
```

### Sublist.id

- **Property Description**
  - Returns the internal ID of the sublist.
**Sublist.isChanged**

**Property Description**
Indicates whether the sublist has changed on the record form.

**Type**
boolean `true` | `false` (read-only)

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/record Module

**Sibling Object Members**
Sublist Object Members

**Since**
2015.2

**Syntax**

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
  type: record.Type.SALES_ORDER,
  id: 275
});

var objSublist = objRecord.getSublist({
  sublistId: 'item'
});
//Perform an action with the objSublist.id value
...
// Add additional code.
```

---

**Type**
string (read-only)

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/record Module

**Sibling Object Members**
Sublist Object Members

**Since**
2015.2

**Syntax**

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
  type: record.Type.SALES_ORDER,
  id: 275
});
```

---
Sublist.isDisplay

**Property Description**
Indicates whether the sublist is displayed on the record form.

**Type**
boolean \( \text{true} \) | \( \text{false} \) (read-only)

**Supported Script Types**
Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/record Module

**Sibling Object Members**
Sublist Object Members

**Since**
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 275
});

var objSublist = objRecord.getSublist({
    sublistId: 'item'
});

if(objSublist.isDisplay){
    //Perform an action when objSublist.isDisplay is true
}
...
// Add additional code.
```

Sublist.type

**Property Description**
Returns the sublist type. For more information on sublist types, see serverWidget.SublistType.
Important: Sublist.type will return a lower case string representing the sublist type. For example, inlineeditor not INLINEEDITOR.

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.load(
    {
        type: record.Type.SALES_ORDER,
        id: 275
    });

var objSublist = objRecord.getSublist(
    {
        sublistId: 'item'
    });

if(objSublist.type === 'inlineeditor'){
    //Perform an action
}
...
// Add additional code.
```

**record.attach(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Attaches a record to another record.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> For the promise version of this method, see record.attach.promise(options). Note that promises are only supported in client scripts.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>void</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>10 units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
</tbody>
</table>
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.record</td>
<td>record.Record</td>
<td>required</td>
<td>The record to attach.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.record.type</td>
<td>string</td>
<td>required</td>
<td>The type of record to attach. Set this value using the record.Type enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.record.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the record to attach.</td>
</tr>
<tr>
<td>options.to</td>
<td>record.Record</td>
<td>required</td>
<td>The record that the options.record gets attached to.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.to.type</td>
<td>string</td>
<td>required</td>
<td>The record type of the record to attach to. Set the value using the record.Type enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.to.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the record to attach to.</td>
</tr>
<tr>
<td>options.attributes</td>
<td>Object</td>
<td>optional</td>
<td>The name-value pairs containing attributes for the attachment. By default, this value is null.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var id = record.attach({
```
record: {
  type: 'file',
  id: '200'
},

to: {
  type: 'customer',
  id: '90'
}

});
...
// Add additional code.

record.attach.promise(options)

Method Description
Attaches a record asynchronously to another record.

**Note:** For information about the parameters and errors thrown for this method, see `record.attach(options)`. For more information about promises, see Promise Object.

Returns
Promise

Supported Script Types
Client-side scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance
10 units

Module
N/record Module

Since
2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.record</td>
<td>record.Record</td>
<td>required</td>
<td>The record to attach.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.record.type</td>
<td>string</td>
<td>required</td>
<td>The type of record to attach. Set the value using the <code>record.Type</code> enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To attach a file from the file cabinet to a record, set type to <code>file</code>.</td>
<td></td>
</tr>
<tr>
<td>options.record.id</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the record to attach.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.to</td>
<td>record.Record</td>
<td>required</td>
<td>The record that the <code>options.record</code> gets attached to.</td>
<td>2015.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>-------------------</td>
<td>-------------------</td>
<td>--------------------</td>
<td>----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>options.to.type</td>
<td>string</td>
<td>required</td>
<td>The record type of the record to attach to.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set the value using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To attach a file from the file cabinet to a record, set type to file.</td>
<td></td>
</tr>
<tr>
<td>options.to.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the record to attach to.</td>
</tr>
<tr>
<td>options.attributes</td>
<td>Object</td>
<td>optional</td>
<td>The name-value pairs containing attributes for the attachment.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
<td></td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code.
...
function attachRecord() {

    var attachRecordPromise = record.attach.promise({
        record: {
            type: record.Type.CONTACT,
            id: '97'
        },
        to: {
            type: record.Type.OPPORTUNITY,
            id: '16'
        }
    });

    attachRecordPromise.then(function() {

        // Add any other needed logic that shouldn't execute until
        // after the contact record is attached to the opportunity.

        log.debug({
            title: 'Record updated',
            details: 'Attachment successful'
        });
    });
```
record.copy(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a new record by copying an existing record in NetSuite.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>For the promise version of this method, see record.copy.promise(options). Note that promises are only supported in client scripts.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>record.Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>Transaction records: 10 usage units</td>
</tr>
<tr>
<td></td>
<td>Custom records: 2 usage units</td>
</tr>
<tr>
<td></td>
<td>All other records: 5 usage units</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/record Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### 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>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the following guidelines:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When copying an instance of a standard NetSuite record type, set this value by using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When copying an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.</td>
<td></td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the existing record instance in NetSuite.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Parameter Details

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>Determines whether the new record is created in dynamic mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to <code>true</code>, the new record is created in dynamic mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to <code>false</code>, the new record is created in standard mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is <code>false</code>.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> For additional information on standard and dynamic mode, see <code>record.Record</code> and SuiteScript 2.0 – Standard and Dynamic Modes.</td>
<td></td>
</tr>
<tr>
<td>options.#defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a list of available record default values, see <code>N/record Default Values</code> in the NetSuite Help Center.</td>
<td></td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
// Add additional code.
...
var objRecord = record.copy({
  type: record.Type.SALES_ORDER,
  id: 157,
  isDynamic: true,
  defaultValues: {
    entity: 107
  }
});
// Add additional code.
```

**record.copy.promise(options)**

**Method Description**

Creates a new record asynchronously by copying an existing record in NetSuite.
### Note:
For information about the parameters and errors thrown for this method, see `record.copy(options)`. For more information on promises, see `Promise Object`.

**Returns**
Promise

**Supported Script Types**
Client-side scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**
Transaction records: 10 usage units
Custom records: 2 usage units
All other records: 5 usage units

**Module**
N/record Module

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the following guidelines:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ When copying an instance of a standard NetSuite record type, set this value by using the <code>record.Type</code> enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ When copying an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.</td>
<td></td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the existing record instance in NetSuite.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether the new record is created in dynamic mode.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ If set to <code>true</code>, the new record is created in dynamic mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ If set to <code>false</code>, the new record is created in standard mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is <code>false</code>.</td>
<td></td>
</tr>
<tr>
<td>options.#defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameter | Type | Required / Optional | Description
--- | --- | --- | ---
 |  |  | By default, this value is null.

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
</table>
| SSS_MISSING_REQD_ARGUMENT | A required argument is missing or undefined.

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
// Add additional code.
...
function copyRecord() {

    // Copy an instance of a standard record type.
    var copyRecordPromise = record.copy.promise({
        type: record.Type.PHONE_CALL,
        id: 165
    });

    // Note: To copy an instance of a custom record type,
    // use the record type’s string ID instead of the record
    // module’s Type enum. For example:
    // type: 'customrecord_feature'

    copyRecordPromise.then(function(recordObject) {

        recordObject.setValue({
            fieldId: 'title',
            value: 'Sprint 5 bug triage'
        });

        recordObject.setValue({
            fieldId: 'message',
            value: 'Please review the PowerPoint prior to the call.'
        });

        var recordId = recordObject.save();

        // Add any other needed logic that shouldn’t execute until
        // after the record is copied.

        log.debug({
            title: 'Record saved',
            details: 'Id of new record: ' + recordId
        });
    });
}
```
record.create(options)

Method Description
Creates a new record.

**Note:** For the promise version of this method, see `record.create.promise(options)`. Note that promises are only supported in client scripts.

Returns
`record.Record`

Supported Script Types
Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

Governance
Transaction records: 10 usage units
Custom records: 2 usage units
All other records: 5 usage units

Module
N/record Module

Since
2015.2

Parameters

**Note:** The options parameter is a JavaScript Object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type. This value determines the <code>Record.type</code> property of the record that is created. This property is read-only on an existing record. Use the following guidelines:</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

- When creating an instance of a standard NetSuite record type, set this value by using the `record.Type` enum.
- When creating an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic [Custom Record](#).
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether the new record is created in dynamic mode.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td>- If set to true, the new record is created in dynamic mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to false, the new record is created in standard mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is false.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** For additional information on standard and dynamic mode, see record.Record and SuiteScript 2.0 – Standard and Dynamic Modes.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a list of available record default values, see N/record Default Values in the NetSuite Help Center.</td>
<td></td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...

// Start the process of creating a sales order record.

var objRecord = record.create(
  
  type: record.Type.SALES_ORDER,
  
  isDynamic: true,
  
  defaultValues: {
    
    entity: 87
  
  }

);

// Start the process of creating an instance of a custom record type.

var customRecord = record.create(
  
  type: 'customrecord_feature',
  
  isDynamic: true

);
```
record.create.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a new record asynchronously.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>For information about the parameters and errors thrown for this method, see record.create(options). For more information on promises, see Promise Object.</td>
</tr>
<tr>
<td>Returns</td>
<td>Promise</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>Transaction records: 10 usage units</td>
</tr>
<tr>
<td></td>
<td>Custom records: 2 usage units</td>
</tr>
<tr>
<td></td>
<td>All other records: 5 usage units</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

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>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value determines the Record.type property of the record that is created. This property is read-only on an existing record.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the following guidelines:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When creating an instance of a standard NetSuite record type, set this value by using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When creating an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.</td>
<td></td>
</tr>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether the new record is created in dynamic mode.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td>- If set to true, the new record is created in dynamic mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td>- If set to false, the new record is created in standard mode.</td>
<td></td>
</tr>
</tbody>
</table>
**N/record Module**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.#defaultValuesObject</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code.
...
function createRecord() {

    // Create an instance of a standard record type.
    var createRecordPromise = record.create.promise({
        type: record.Type.PHONE_CALL,
        isDynamic: true
    });

    // Note: To create an instance of a custom record type, use the record type's string ID instead of the record module's Type enum. For example:
    // type: 'customrecord_feature'

    createRecordPromise.then(function(objRecord) {
        objRecord.setValue({
            fieldId: 'title',
            value: 'sprint planning'
        });

        var recordId = objRecord.save();

        log.debug({
            recordId: recordId
        });
    });
}
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.
record.delete(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Deletes a record.</th>
</tr>
</thead>
</table>

**Note:** For the promise version of this method, see `record.delete.promise(options)`. Note that promises are only supported in client scripts.

**Returns**
The internal ID of the deleted `record.Record`.

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**
Transaction records: 20 usage units
Custom records: 4 usage units
All other records: 10 usage units

**Module**
N/record Module

**Since**
2015.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type. Use the following guidelines:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When deleting an instance of a standard NetSuite record type, set this value by using the <code>record.Type</code> enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When deleting an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic <a href="#">Custom Record</a>.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### 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>options.id</td>
<td>number</td>
<td>string</td>
<td>The internal ID of the record instance to be deleted.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...

// Delete a sales order.

var salesOrderRecord = record.delete({
    type: record.Type.SALES_ORDER,
    id: 88,
});

// Delete an instance of a custom record type with the ID customrecord_feature.

var featureRecord = record.delete({
    type: 'customrecord_feature',
    id: 3,
});
...

// Add additional code.
```

**record.delete.promise(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Deletes a record asynchronously.</th>
</tr>
</thead>
</table>

**Note:** For information about the parameters and errors thrown for this method, see `record.delete(options)`. For more information on promises, see [Promise Object](#).

<table>
<thead>
<tr>
<th>Returns</th>
<th>Promise</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>Transaction records: 20 usage units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Custom records: 4 usage units</td>
</tr>
</tbody>
</table>
All other records: 10 usage units

Module N/record Module
Since 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type. Use the following guidelines:</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ When deleting an instance of a standard NetSuite record type, set this value by using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>■ When deleting an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.</td>
<td></td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the record instance to be deleted.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
// Add additional code.

// Delete an instance of a standard NetSuite record
...

function deleteRecord() {

    var deleteRecordPromise = record.delete.promise({
        type: record.Type.PHONE_CALL,
        id: 109
    });

    // To delete an instance of a custom record type, use
    // the string ID in the type field. For example:
    // type: 'customrecord_feature'

    deleteRecordPromise.then(function() {
```
record.detach(options)

Method Description: Detaches a record from another record.

**Note:** For the promise version of this method, see `record.detach.promise(options)`. Note that promises are only supported in client scripts.

Returns: void

Supported Script Types: Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

Governance: 10 units

Module: N/record Module

Since: 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.record</td>
<td>record.Record</td>
<td>required</td>
<td>The record to be detached.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.record.type</td>
<td>string</td>
<td>required</td>
<td>The type of record to be detached.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set this value using the <code>record.Type</code> enum.</td>
<td></td>
</tr>
<tr>
<td>options.record.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The ID of the record to be detached.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------</td>
</tr>
<tr>
<td>options.from</td>
<td>record.Record</td>
<td>required</td>
<td>The destination record that options.record should be detached from.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.from.type</td>
<td>string</td>
<td>required</td>
<td>The type of the destination.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set this value using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td>options.from.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The ID of the destination.</td>
</tr>
<tr>
<td>options.attributes</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
<td></td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code.
...
record.detach({
  record: {
    type: 'file',
    id: '200'
  },
  from: {
    type: 'customer',
    id: '90'
  }
})
...
// Add additional code.
```

---

**record.detach.promise(options)**

**Method Description**

Detaches a record from another record asynchronously.

**Note:** For information about the parameters and errors thrown for this method, see `record.detach(options)`. For more information on promises, see `Promise Object`.  

**Returns**

Promise
Supported Script Types  Client-side scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance  10 units

Module  N/record Module

Since  2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.record</td>
<td>record.Record</td>
<td>required</td>
<td>The record to be detached.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.record.type</td>
<td>string</td>
<td>required</td>
<td>The type of record to be detached.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set this value using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td>options.record.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The ID of the record to be detached.</td>
</tr>
<tr>
<td>options.from</td>
<td>record.Record</td>
<td>required</td>
<td>The destination record that options.record should be detached from.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.from.type</td>
<td>string</td>
<td>required</td>
<td>The type of the destination.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set this value using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td>options.from.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The ID of the destination.</td>
</tr>
<tr>
<td>options.attributes</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
// Add additional code.
```
function detachRecord() {
    var detachRecordPromise = record.detach.promise({
        record: {
            type: record.Type.CONTACT,
            id: '98'
        },
        from: {
            type: record.Type.OPPORTUNITY,
            id: '16'
        }
    });
    detachRecordPromise.then(function() {
        // Add any other needed logic that shouldn’t execute until
        // after the contact record is detached from the opportunity.
        log.debug({
            title: 'Record updated',
            details: 'Contact record detached'
        });
    }, function(e) {
        log.error({
            title: e.name,
            details: e.message
        });
    });
    ...
    // Add additional code.

record.load(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Loads an existing record.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> For the promise version of this method, see record.load.promise(options). Note that promises are only supported in client scripts. Make sure to save the record before loading it.</td>
<td></td>
</tr>
<tr>
<td>Returns</td>
<td>record.Record</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>Transaction records: 10 usage units</td>
</tr>
<tr>
<td></td>
<td>Custom records: 2 usage units</td>
</tr>
<tr>
<td></td>
<td>All other records: 5 usage units</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
</tbody>
</table>
Since 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type. Use the following guidelines:</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When loading an instance of a standard NetSuite record type, set this value by using the record.Type enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When loading an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.</td>
<td></td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the existing record instance in NetSuite. The internal ID of the record is displayed on the list page for the record type.</td>
<td></td>
</tr>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether the record is loaded in dynamic mode. By default, this value is false.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to <code>true</code>, the record is loaded in dynamic mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to <code>false</code>, the record is loaded in standard mode.</td>
<td></td>
</tr>
<tr>
<td>options.#defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a list of available record default values, see N/record Default Values in the NetSuite Help Center.</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...

// Load a sales order.
var objRecord = record.load({
    type: record.Type.SALES_ORDER,
    id: 157,
    isDynamic: true,
});

// Load an instance of a custom record type with the ID customrecord_feature.

var newFeatureRecord = record.load({
    type: 'customrecord_feature',
    id: 1,
    isDynamic: true
});
...
// Add additional code.
```

**record.load.promise(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Loads an existing record asynchronously.</th>
</tr>
</thead>
</table>

**Note:** For information about the parameters and errors thrown for this method, see `record.load(options)`. For more information on promises, see [Promise Object](#).

<table>
<thead>
<tr>
<th>Returns</th>
<th>Promise</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Client Script Type</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>Transaction records: 10 usage units</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Custom records: 2 usage units</td>
</tr>
<tr>
<td></td>
<td>All other records: 5 usage units</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/record Module</th>
</tr>
</thead>
</table>

| Since                 | 2015.2 |
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>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type.  Needs to be set when creating a new record. For more information, see record.Type.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the existing record instance in NetSuite.</td>
<td></td>
</tr>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether the record is loaded in dynamic mode.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.#defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
function loadRecord() {
    // Code implementation goes here.
}
```
// Load an instance of a standard NetSuite record
// type.

var loadRecordPromise = record.load.promise({
    type: record.Type.PHONE_CALL,
    id: 712
});

// Note: To load an instance of a custom record type,
// use the record type's string ID. For example:
// type: 'customrecord_feature'

loadRecordPromise.then(function(objRecord) {
    objRecord.setValue({
        fieldId: 'message',
        value: 'We will start the call with a retrospective.'
    });
    var recordId = objRecord.save();

    // Add any other needed logic that shouldn't execute
    // until after the record is instantiated.

    log.debug({
        title: 'Record updated',
        details: 'Updated record ID: ' + recordId
    });
}, function(e) {
    log.error({
        title: 'Unable to load record',
        details: e.name
    });
});

// Add additional code.

record.submitFields(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Updates and submits one or more body fields on an existing record in NetSuite, and returns the internal ID of the parent record.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>When you use this method, you do not need to load or submit the parent record.</td>
</tr>
<tr>
<td></td>
<td>You can use this method to edit and submit the following:</td>
</tr>
<tr>
<td></td>
<td>■ Standard body fields that support inline editing (direct list editing). For more information, see the help topic Using Inline Editing.</td>
</tr>
<tr>
<td></td>
<td>■ Custom body fields that support inline editing.</td>
</tr>
<tr>
<td></td>
<td>You cannot use this method to edit and submit the following:</td>
</tr>
<tr>
<td></td>
<td>■ Select fields</td>
</tr>
<tr>
<td></td>
<td>■ Sublist line item fields</td>
</tr>
</tbody>
</table>
Subrecord fields (for example, address fields)

**Note:** For the promise version of this method, see `record.submitFields.promise(options)`. Note that promises are only supported in client scripts.

<table>
<thead>
<tr>
<th>Returns</th>
<th>The internal ID of the parent record.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>Transaction records: 10 usage units</td>
</tr>
<tr>
<td></td>
<td>Custom records: 2 usage units</td>
</tr>
<tr>
<td></td>
<td>All other records: 5 usage units</td>
</tr>
<tr>
<td>Module</td>
<td>N/record Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Use the following guidelines:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When working with an instance of a standard NetSuite record type, set this value by using the <code>record.Type</code> enum.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- When working with an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.</td>
<td></td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the existing record instance in NetSuite.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.values</td>
<td>Object</td>
<td>required</td>
<td>The ID-value pairs for each field you want to edit and submit.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.options</td>
<td>Object</td>
<td>optional</td>
<td>Additional options to set for the record.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
## 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>options.**options.**enableSourcing</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether to enable sourcing during the record update. By default, this value is true.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.**options.**ignoreMandatoryFields</td>
<td>boolean</td>
<td>optional</td>
<td>Indicates whether to ignore mandatory fields during record submission. By default, this value is false.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

## Syntax

```javascript
// Add additional code.
...

// Submit a new value for a sales order's memo field.

var id = record.submitFields({
  type: record.Type.SALES_ORDER,
  id: 1,
  values: {
    memo: 'ABC'
  },
  options: {
    enableSourcing: false,
    ignoreMandatoryFields : true
  }
});

// Submit a new value for a field on an instance of the 'customrecord_book' custom record type.

var otherId = record.submitFields({
  type: 'customrecord_book',
  id: '4',
  values: {
    'custrecord_rating': '2'
  }
});

... // Add additional code.
```
record.submitFields.promise(options)

### Method Description
Updates and submits one or more body fields asynchronously on an existing record in NetSuite, and returns the internal ID of the parent record.

When you use this method, you do not need to load or submit the parent record.

You can use this method to edit and submit the following:

- Standard body fields that support inline editing (direct list editing). For more information, see the help topic Using Inline Editing.
- Custom body fields that support inline editing.

You cannot use this method to edit and submit the following:

- Select fields
- Sublist line item fields
- Subrecord fields (for example, address fields)

**Note:** For information about the parameters and errors thrown for this method, see `record.submitFields(options)`. For more information on promises, see Promise Object.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Promise</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client-side scripts</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
<td></td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>Transaction records: 10 usage units</td>
</tr>
<tr>
<td></td>
<td>Custom records: 2 usage units</td>
</tr>
<tr>
<td></td>
<td>All other records: 5 usage units</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/record Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The record type.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Use the following guidelines:

- When working with an instance of a standard NetSuite record type, set this value by using the `record.Type` enum.
- When working with an instance of a custom record type, set this value by using the custom record type's string ID. For help finding this ID, see the help topic Custom Record.
### 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>options.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the existing record instance in NetSuite.</td>
</tr>
<tr>
<td>options.values</td>
<td>Object</td>
<td>required</td>
<td></td>
<td>The ID-value pairs for each field you want to edit and submit.</td>
</tr>
<tr>
<td>options.options</td>
<td>Object</td>
<td>optional</td>
<td></td>
<td>Additional options to set for the record.</td>
</tr>
<tr>
<td>options.options.<strong>enablesourcing</strong></td>
<td>boolean</td>
<td></td>
<td>true</td>
<td>Indicates whether to enable sourcing during the record update. By default, this value is true.</td>
</tr>
<tr>
<td>options.options.<strong>ignoreMandatoryFields</strong></td>
<td>boolean</td>
<td></td>
<td>true</td>
<td>Indicates whether to ignore mandatory fields during record submission. By default, this value is false.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
// Add additional code
...
function submitFields() {
  var submitFieldsPromise = record.submitFields.promise({
    type: record.Type.PHONE_CALL,
    id: 171,
    values: {
      title: 'Sprint 3 planning'
    },
  });

  submitFieldsPromise.then(function(recordId) {
    // Add any needed logic that shouldn’t execute until
    // after the new value is submitted.
    log.debug({
      title: 'Record updated',
      details: 'Id of updated record: ' + recordId
    });
  }, function(e) {
    log.error({
      title: e.name,
    });
  });
}
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.
record.transform(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Transforms a record from one type into another, using data from an existing record. You can use this method to automate order processing, creating item fulfillment transactions and invoices off of orders. For a list of supported transformations, see Supported Transformation Types.</th>
</tr>
</thead>
</table>

**Note:** For the promise version of this method, see record.transform.promise(options). Note that promises are only supported in client scripts.

**Returns**

record.Record

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

Transaction records: 10 usage units
Custom records: 2 usage units
All other record types: 5 usage units

**Module**

N/record Module

**Since**

2015.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fromType</td>
<td>string</td>
<td>required</td>
<td>The record type of the existing record instance being transformed. This value sets the Record.type property for the record. This property is read-only and cannot be changed after the record is loaded. Set this value using the record.Type.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.fromId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the existing record instance being transformed.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.toType</td>
<td>string</td>
<td>required</td>
<td>The record type of the record returned when the transformation is complete.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether the new record is created in dynamic mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to <code>true</code>, the new record is created in dynamic mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- If set to <code>false</code>, the new record is created in standard mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is <code>false</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> For additional information on standard and dynamic mode, see <code>record.Record</code> and SuiteScript 2.0 – Standard and Dynamic Modes.</td>
</tr>
<tr>
<td>options.#defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For a list of available record default values, see N/record Default Values in the NetSuite Help Center.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```javascript
// Add additional code.
...
var objRecord = record.transform({
  fromType: record.Type.CUSTOMER,
  fromId: 107,
  toType: record.Type.SALES_ORDER,
  isDynamic: true,
});
...
// Add additional code.
```

### Supported Transformation Types

<table>
<thead>
<tr>
<th>Original Record Type</th>
<th>Transformed Record Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Build/Assembly</td>
<td>Assembly Build</td>
</tr>
<tr>
<td>Assembly Build</td>
<td>Assembly Unbuild</td>
</tr>
<tr>
<td>Original Record Type</td>
<td>Transformed Record Type</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>Cash Sale</td>
<td>Cash Sale</td>
</tr>
<tr>
<td>Customer</td>
<td>Cash Sale</td>
</tr>
<tr>
<td>Customer</td>
<td>Customer Payment</td>
</tr>
<tr>
<td>Customer</td>
<td>Quote</td>
</tr>
<tr>
<td>Customer</td>
<td>Invoice</td>
</tr>
<tr>
<td>Customer</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Employee</td>
<td>Sales Order</td>
</tr>
<tr>
<td>Employee</td>
<td>Expense Report</td>
</tr>
<tr>
<td>Quote</td>
<td>Time</td>
</tr>
<tr>
<td>Quote</td>
<td>Cash Sale</td>
</tr>
<tr>
<td>Quote</td>
<td>Invoice</td>
</tr>
<tr>
<td>Quote</td>
<td>Sales Order</td>
</tr>
<tr>
<td>Invoice</td>
<td>Credit Memo</td>
</tr>
<tr>
<td>Invoice</td>
<td>Customer Payment</td>
</tr>
<tr>
<td>Invoice</td>
<td>Return Authorization</td>
</tr>
<tr>
<td>Lead</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Cash Sale</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Quote</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Invoice</td>
</tr>
<tr>
<td>Opportunity</td>
<td>Sales Order</td>
</tr>
<tr>
<td>Prospect</td>
<td>Quote</td>
</tr>
<tr>
<td>Prospect</td>
<td>Opportunity</td>
</tr>
<tr>
<td>Prospect</td>
<td>Sales Order</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>Item Receipt</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>Vendor Bill</td>
</tr>
<tr>
<td>Purchase Order</td>
<td>Vendor Return Authorization</td>
</tr>
<tr>
<td>Return Authorization</td>
<td>Cash Refund</td>
</tr>
<tr>
<td>Return Authorization</td>
<td>Credit Memo</td>
</tr>
<tr>
<td>Return Authorization</td>
<td>Item Receipt</td>
</tr>
<tr>
<td>Return Authorization</td>
<td>Revenue Commitment Reversal</td>
</tr>
</tbody>
</table>
record.transform.promise(options)

### Method Description
Transforms a record from one type into another asynchronously, using data from an existing record.

You can use this method to automate order processing, creating item fulfillment transactions and invoices off of orders.

For a list of supported transformations, see [Supported Transformation Types](#).

**Note:** For information about the parameters and errors thrown for this method, see `record.transform(options)`. For more information on promises, see [Promise Object](#).

### Returns
Promise

### Supported Script Types
Client-side scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

### Governance
Transaction records: 10 usage units
Custom records: 2 usage units
All other record types: 5 usage units
**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.fromType</td>
<td>string</td>
<td>required</td>
<td>The record type of the existing record instance being transformed.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This value sets the <code>Record.type</code> property for the record. This property is read-only and cannot be changed after the record is loaded.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Set this value using the <code>record.Type</code>.</td>
<td></td>
</tr>
<tr>
<td>options.fromId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the existing record instance being transformed.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.toType</td>
<td>string</td>
<td>required</td>
<td>The record type of the record returned when the transformation is complete.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.isDynamic</td>
<td>boolean</td>
<td></td>
<td>Determines whether the new record is created in dynamic mode.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>optional</td>
<td>If set to <code>true</code>, the new record is created in dynamic mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td>If set to <code>false</code>, the new record is created in standard mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is <code>false</code>.</td>
<td></td>
</tr>
<tr>
<td>options.@defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>Name-value pairs containing default values of fields in the new record.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is null.</td>
<td></td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required argument is missing or undefined.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
// Add additional code.
```
function transformRecord() {

    var transformRecordPromise = record.transform.promise({
        fromType: record.Type.QUOTE,
        fromId: 25,
        toType: record.Type.SALES_ORDER,
        isDynamic: true,
    });

    transformRecordPromise.then(function(recordObject) {

        var recordId = recordObject.save();

        // Add any other needed logic that shouldn’t execute until
        // after the record is transformed.

        log.debug({
            title: 'Record saved',
            details: 'Id of new record: ' + recordId
        });
    }, function(e) {

        log.error({
            title: e.name,
            details: e.message
        });
    });

    // Add additional code.

}

record.Type

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for supported record types.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This enum is used to set the value of the Record.type property in cases where you are working with an instance of a standard NetSuite record type. (If you are working with an instance of a custom record type, you set the Record.type property by using the custom record type’s string ID. For more help finding this ID, see the help topic Custom Record.)</td>
</tr>
<tr>
<td>Note:</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/record Module

Since

2015.2
### Values

<table>
<thead>
<tr>
<th>ACCOUNT</th>
<th>EMAIL_TEMPLATE</th>
<th>POSITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACCOUNTING_BOOK</td>
<td>EMPLOYEE</td>
<td>PRICE_BOOK</td>
</tr>
<tr>
<td>ACCOUNTING_CONTEXT</td>
<td>ENTITY_AREA_ACCOUNT_AREA_MAPPING</td>
<td>PRICE_LEVEL</td>
</tr>
<tr>
<td>ACCOUNTING_PERIOD</td>
<td>ESTIMATE</td>
<td>PRICE_PLAN</td>
</tr>
<tr>
<td>ADVANCED_INTERCOMPANY_TRANSACTION</td>
<td>ROUTING_CATEGORY</td>
<td>PRICING_GROUP</td>
</tr>
<tr>
<td>ALLOCATION_SCHEDULE</td>
<td>EXPENSE_REPORT</td>
<td>PROJECT_EXPENSE_TYPE</td>
</tr>
<tr>
<td>AMORTIZATION_AREA_SCHED</td>
<td>FAIR_VALUE_PRICE</td>
<td>PROJECT_TASK</td>
</tr>
<tr>
<td>AMORTIZATION_AREA_TEMPLATE</td>
<td>FIXED_AREA_AMOUNT_AREA_PROJECT_AREA_COMMITMENT</td>
<td>REVENUE_AREA_COMMITMENT_REVERSAL</td>
</tr>
<tr>
<td>ASSEMBLY_BUILD</td>
<td>FOLDER</td>
<td>PROMOTION_CODE</td>
</tr>
<tr>
<td>ASSEMBLY_ITEM</td>
<td>FULFILLMENT_REQUEST</td>
<td>PROSPECT</td>
</tr>
<tr>
<td>ASSEMBLY_UNBUILD</td>
<td>GENERIC_RESOURCE</td>
<td>PURCHASE_CONTRACT</td>
</tr>
<tr>
<td>BILLING_ACCOUNT</td>
<td>GIFT_CERTIFICATE</td>
<td>PURCHASE_ORDER</td>
</tr>
<tr>
<td>BILLING_CLASS</td>
<td>GIFT_CERTIFICATE_ITEM</td>
<td>PURCHASE_REQUISITION</td>
</tr>
<tr>
<td>BILLING_RATE_CARD</td>
<td>GLOBAL_AREA_ACCOUNT_AREA_MAPPING</td>
<td>RATE_PLAN</td>
</tr>
<tr>
<td>BILLING_REVENUE_EVENT</td>
<td>GLOBAL_AREA_INVENTORY_AREA_RELATIONSHIP</td>
<td>REVERSE_ALLOCATE_ITEM</td>
</tr>
<tr>
<td>BILLING_SCHEDULE</td>
<td>GOVERNMENT_AREA_ID_AREA_ID</td>
<td>RECEIVE_AREA_INBOUND_AREA_SHIPMENT</td>
</tr>
<tr>
<td>BIN</td>
<td>HCM_JOB</td>
<td>RESOURCE_ALLOCATION</td>
</tr>
<tr>
<td>BIN_TRANSFER</td>
<td>INBOUND_SHIPMENT</td>
<td>RESTLET</td>
</tr>
<tr>
<td>BIN_WORKSHEET</td>
<td>INTERCOMP_AREA_ALLOCATION_AREA_SCHED</td>
<td>RETURN_AREA_AUTHORIZATION</td>
</tr>
<tr>
<td>BLANKET_AREA_PURCHASE_AREA_ORDER</td>
<td>INTERVAL_AREA_COMPANY_AREA_JOURNAL_AREA_COMMITMENT</td>
<td>REVENUE_AREA_COMMITMENT</td>
</tr>
<tr>
<td>BOM</td>
<td>INVENTORY_ADJUSTMENT</td>
<td>REVENUE_AREA_COMMITMENT_REVERSAL</td>
</tr>
<tr>
<td>BOM_REVISION</td>
<td>INVENTORY_ACCOUNT</td>
<td>REVENUE_PLAN</td>
</tr>
<tr>
<td>BULK_AREA_OWNERSHIP_AREA_TRANSFER</td>
<td>INVENTORY_AREA_COST_AREA_REVALUATION</td>
<td>REV_REC_SCHEDULE</td>
</tr>
<tr>
<td>BUNDLE_AREA_INSTALLATION_AREA_SCRIPT</td>
<td>INVENTORY_COUNT</td>
<td>REV_REC_TEMPLATE</td>
</tr>
<tr>
<td>CALENDAR_EVENT</td>
<td>INVENTORY_DETAIL</td>
<td>SALES_ORDER</td>
</tr>
<tr>
<td>CAMPAIGN</td>
<td>INVENTORY_ITEM</td>
<td>SALES_TAX_ITEM</td>
</tr>
<tr>
<td>CAMPAIGN_RESPONSE</td>
<td>INVENTORY_NUMBER</td>
<td>SCHEDULED_SCRIPT</td>
</tr>
<tr>
<td>CAMPAIGN_TEMPLATE</td>
<td>INVENTORY_TRANSFER</td>
<td>SCHEDULED_AREA_SCRIPT_AREA_INSTANCE</td>
</tr>
<tr>
<td>CASH_REFUND</td>
<td>INVOICE</td>
<td>SCRIPT_DEPLOYMENT</td>
</tr>
<tr>
<td>CASH_SALE</td>
<td>ISSUE</td>
<td>SERIALIZED_AREA.Assembly_AREA_ITEM</td>
</tr>
<tr>
<td>CHARGE</td>
<td>ISSUE_PRODUCT</td>
<td>SERVICES</td>
</tr>
<tr>
<td>CHARGE_RULE</td>
<td>ISSUE_PRODUCT_VERSION</td>
<td>SERIALIZE</td>
</tr>
<tr>
<td>CHECK</td>
<td>ITEM_ACCOUNT_MAPPING</td>
<td>SERVICE_ITEM</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>ITEM_DEMAND_PLAN</td>
<td>SHIP_ITEM</td>
</tr>
<tr>
<td>CLIENT_SCRIPT</td>
<td>ITEM_FULFILLMENT</td>
<td>SOLUTION</td>
</tr>
<tr>
<td>CMS_CONTENT</td>
<td>ITEM_GROUP</td>
<td>STORE_AREA_PICKUP_AREA_FULFILLMENT</td>
</tr>
<tr>
<td>CMS_CONTENT_TYPE</td>
<td>ITEM_AREA_LOCATION_AREA_CONFIGURATION</td>
<td>SUBSCRIPTION</td>
</tr>
<tr>
<td>COMMERCE_CATEGORY</td>
<td>ITEM_RECEIPT</td>
<td>SUBSCRIPTION_AREA.change_AREA_ORDER</td>
</tr>
<tr>
<td>COMPETITOR</td>
<td>ITEM_REVISION</td>
<td>SUBSCRIPTION_LINE</td>
</tr>
<tr>
<td>CONSOLIDATED_AREA_EXCHANGE_AREA_RATE</td>
<td>ITEM_SUPPLY_PLAN</td>
<td>SUBSCRIPTION_PLAN</td>
</tr>
<tr>
<td>CONTACT</td>
<td>JOB</td>
<td>SUBSIDIARY</td>
</tr>
<tr>
<td>CONTACT_CATEGORY</td>
<td>JOB_REQUISITION</td>
<td>SUPPORT_AREA</td>
</tr>
<tr>
<td>CONTACT_ROLE</td>
<td>JOB_STATUS</td>
<td>SUPPORT_CASE</td>
</tr>
<tr>
<td>COST_CATEGORY</td>
<td>JOB_TYPE</td>
<td>SUPPORT_CASE</td>
</tr>
<tr>
<td>Member Type</td>
<td>Script Type</td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>CREDIT_CARD_CHARGE</td>
<td>CREDIT_CARD_REFUND</td>
<td></td>
</tr>
<tr>
<td>CREDIT_MEMO</td>
<td>CURRENCY</td>
<td></td>
</tr>
<tr>
<td>CUSTOMER</td>
<td>CUSTOMER_CATEGORY</td>
<td></td>
</tr>
<tr>
<td>CUSTOMER_DEPOSIT</td>
<td>CUSTOMER_MESSAGE</td>
<td></td>
</tr>
<tr>
<td>CUSTOMER_PAYMENT</td>
<td>CUSTOMER_PAYMENT_AUTHORIZATION</td>
<td></td>
</tr>
<tr>
<td>CUSTOMER_REFUND</td>
<td>CUSTOMER_STATUS</td>
<td></td>
</tr>
<tr>
<td>CUSTOMER_SUBSIDIARY_RELATIONSHIP</td>
<td>CUSTOM_RECORD</td>
<td></td>
</tr>
<tr>
<td>CUSTOM_TRANSACTION</td>
<td>DEPARTMENT</td>
<td></td>
</tr>
<tr>
<td>DEPOSIT</td>
<td>DEPOSIT_APPLICATION</td>
<td></td>
</tr>
<tr>
<td>DESCRIPTION_ITEM</td>
<td>DISCOUNT_ITEM</td>
<td></td>
</tr>
<tr>
<td>DOWNLOAD_ITEM</td>
<td>Download Item</td>
<td></td>
</tr>
<tr>
<td>KIT_ITEM</td>
<td>KUDOS</td>
<td></td>
</tr>
<tr>
<td>LABOR_BASED_PROJECT_REVENUE_RULE</td>
<td>LEAD</td>
<td></td>
</tr>
<tr>
<td>LOCATION</td>
<td>LOT_NUMBERED.Assembly_Item</td>
<td></td>
</tr>
<tr>
<td>LOT_NUMBERED.Inventory_Item</td>
<td>MANUFACTURING.COST TEMPLATE</td>
<td></td>
</tr>
<tr>
<td>MANUFACTURING.OPERATION_TASK</td>
<td>MAP_REDUCE_SCRIPT</td>
<td></td>
</tr>
<tr>
<td>MAPUP_ITEM</td>
<td>MESSAGE</td>
<td></td>
</tr>
<tr>
<td>MESSAGE</td>
<td>MFG_PLANNED_TIME</td>
<td></td>
</tr>
<tr>
<td>NEXUS</td>
<td>NON_INVENTORY_ITEM</td>
<td></td>
</tr>
<tr>
<td>NOTE</td>
<td>NOTE_TYPE</td>
<td></td>
</tr>
<tr>
<td>OPPORTUNITY</td>
<td>ORDER_SCHEDULE</td>
<td></td>
</tr>
<tr>
<td>ORGANIZATION_VALUE</td>
<td>OTHER_CHARGE_ITEM</td>
<td></td>
</tr>
<tr>
<td>OTHER. GOVERNMENT_ISSUED_ID</td>
<td>OTHER_NAME</td>
<td></td>
</tr>
<tr>
<td>PARTNER</td>
<td>PARTNER_CATEGORY</td>
<td></td>
</tr>
<tr>
<td>PASSPORT</td>
<td>PAYCHECK_JOURNAL</td>
<td></td>
</tr>
<tr>
<td>PAYCHECK_JOURNAL</td>
<td>PAYMENT_ITEM</td>
<td></td>
</tr>
<tr>
<td>PAYMENT_ITEM</td>
<td>PAYMENT_METHOD</td>
<td></td>
</tr>
<tr>
<td>PAYROLL_ITEM</td>
<td>PERIOD_END_JOURNAL</td>
<td></td>
</tr>
<tr>
<td>PHONE_CALL</td>
<td>PORTLET</td>
<td></td>
</tr>
<tr>
<td>TASK</td>
<td>TAX_ACCT</td>
<td></td>
</tr>
<tr>
<td>TAX_GROUP</td>
<td>TAX_PERIOD</td>
<td></td>
</tr>
<tr>
<td>TAX_TYPE</td>
<td>TERM</td>
<td></td>
</tr>
<tr>
<td>TERMINATION_REASON</td>
<td>TIME_BILL</td>
<td></td>
</tr>
<tr>
<td>TIME_OFF_CHANGE</td>
<td>TIME_OFF_PLAN</td>
<td></td>
</tr>
<tr>
<td>TIME_OFF_REQUEST</td>
<td>TIME_OFF_RULE</td>
<td></td>
</tr>
<tr>
<td>TIME_OFF_TYPE</td>
<td>TOPIC</td>
<td></td>
</tr>
<tr>
<td>TRANSFER_ORDER</td>
<td>UNITS_TYPE</td>
<td></td>
</tr>
<tr>
<td>USAGE</td>
<td>USEREVENT_SCRIPT</td>
<td></td>
</tr>
<tr>
<td>VENDOR</td>
<td>VENDOR BILL</td>
<td></td>
</tr>
<tr>
<td>VENDOR_CATEGORY</td>
<td>VENDOR CREDIT</td>
<td></td>
</tr>
<tr>
<td>VENDOR_PAYMENT</td>
<td>VENDOR_RETURN_AUTHORIZATION</td>
<td></td>
</tr>
<tr>
<td>WEBSITE</td>
<td>WORKFLOW_ACTION_SCRIPT</td>
<td></td>
</tr>
<tr>
<td>WORK_ORDER</td>
<td>WORK_ORDER_CLOSE</td>
<td></td>
</tr>
<tr>
<td>WORK_ORDER_COMPLETION</td>
<td>WORK_ORDER_ISSUE</td>
<td></td>
</tr>
<tr>
<td>WORKPLACE</td>
<td>WORKPLACE</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/record Module Script Samples](#).

```javascript
// Add additional code.
...
var objRecord = record.delete({
```
N/redirect Module

Use the redirect module to customize navigation within NetSuite by setting up a redirect URL that resolves to a NetSuite resource or external URL. You can redirect users to one of the following:

- URL
- Suitelet
- Record
- Task link
- Saved search
- Unsaved search

**Note:** Suitelets, beforeLoad user events, and synchronous afterSubmit user events are supported. This module does not support beforeSubmit and asynchronous afterSubmit user events.

### N/redirect Module Members

### N/redirect Module Script Sample

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>redirect.###redirect###(options)</td>
<td>void</td>
<td>Suitelets, beforeLoad user events, and synchronous afterSubmit user events</td>
<td>Redirects to the URL of a Suitelet that is available externally (available without login).</td>
</tr>
<tr>
<td></td>
<td>redirect.###toRecord###(options)</td>
<td>void</td>
<td>Suitelets, beforeLoad user events, and synchronous afterSubmit user events</td>
<td>Redirects to a NetSuite record.</td>
</tr>
<tr>
<td></td>
<td>redirect.###toSavedSearch###(options)</td>
<td>void</td>
<td>afterSubmit user events</td>
<td>Redirects to a saved search.</td>
</tr>
<tr>
<td></td>
<td>redirect.###toSavedSearch###(options)</td>
<td>void</td>
<td>afterSubmit user events</td>
<td>Redirects to a saved search result.</td>
</tr>
<tr>
<td></td>
<td>redirect.###toSearch###(options)</td>
<td>void</td>
<td>afterSubmit user events</td>
<td>Redirects to search.</td>
</tr>
<tr>
<td></td>
<td>redirect.###toSearchResult###(options)</td>
<td>void</td>
<td>afterSubmit user events</td>
<td>Redirects to search results.</td>
</tr>
<tr>
<td></td>
<td>redirect.###toSuitelet###(options)</td>
<td>void</td>
<td>Suitelets, beforeLoad user events, and synchronous afterSubmit user events</td>
<td>Redirects to a Suitelet.</td>
</tr>
</tbody>
</table>
### N/redirect Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>redirect.toTaskLink</td>
<td>void</td>
<td>Suitelets, beforeLoad user events, and synchronous afterSubmit user events</td>
<td>Redirects to a tasklink.</td>
</tr>
</tbody>
</table>

### N/redirect Module Script Sample

The following example sets the redirect URL to a newly created task record. To set the redirect using the record id, the record must have been previously submitted.

**Note:** This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/record', 'N/redirect'],
function(record, redirect) {
    function redirectToTaskRecord() {
        var taskTitle = 'New Opportunity';
        var taskRecord = record.create({
            type: record.Type.TASK
        });
        taskRecord.setValue('title', taskTitle);
        var taskRecordId = taskRecord.save();
        redirect.toRecord({
            type: record.Type.TASK,
            id: taskRecordId
        });
    }
    redirectToTaskRecord();
});
```

### redirect.redirect(options)

**Method Description**

Method used to set the redirect to the URL of a Suitelet that is available externally (Suitelets set to Available Without Login on the Script Deployment page).

<table>
<thead>
<tr>
<th>Returns</th>
<th>Void</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Suitelets, beforeLoad user events, and synchronous afterSubmit user events</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/redirect Module</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The URL of a Suitelet that is available externally</td>
</tr>
<tr>
<td>options.parameters</td>
<td>Object</td>
<td>optional</td>
<td>Contains additional URL parameters as key/value pairs.</td>
</tr>
</tbody>
</table>

**Note:** For an external URL, Available without Login must be enabled on the Script Deployment page for the Suitelet.

Syntax

```
//Add additional code
...
redirect.redirect({
    url: /app/site/hosting/scriptlet.nl?script=130&deploy=1,
    parameters: {'custparam_test': 'helloWorld'}
});
...
//Add additional code
```

**redirect.toRecord(options)**

**Method Description**

Method used to set the redirect URL to a specific NetSuite record.

**Note:** If you redirect a user to a record, the record must first exist in NetSuite. If you want to redirect a user to a new record, you must first create and submit the record before redirecting them. You must also ensure that any required fields for the new record are populated before submitting the record.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal id of the target record.</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The type of record.</td>
</tr>
<tr>
<td>options.isEditMode</td>
<td>boolean</td>
<td>optional</td>
<td>Determines whether to return a URL for the record in edit mode or view mode. If set to true, returns the URL to an existing record in edit mode. The default value is false – returns the URL to a record in view mode.</td>
</tr>
<tr>
<td>options.parameters</td>
<td>Object</td>
<td>optional</td>
<td>Contains additional URL parameters as key/value pairs.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
// Add additional code
...
redirect.toRecord({
  type : record.Type.TASK,
  id : taskRecordId,
  parameters: {'custparam_test':'helloWorld'}
});
...
// Add additional code
```

redirect.toSavedSearch(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to load an existing saved search and redirect to the populated search definition page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>afterSubmit user event scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 User Event Script Type.</td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/redirect Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.id  | number | required             | Internal ID of the search. The internal ID is available only when the search is either loaded with `search.load(options)` or after is has been saved with `Search.save()`.
|             |        |                      | Typical values are 55 or 234 or 87, not a value like `customsearch_mysearch`. Any ID prefixed with `customsearch` is a script ID, not the internal system ID for a search. |

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/redirect Module Script Sample.

```javascript
//Add additional code
...
redirect.toSavedSearch({id: 234});
...
//Add additional code
```

### redirect.toSavedSearchResult(options)

**Method Description**
Method used to redirect a user to a search results page for an existing saved search.

**Returns**
Void

**Supported Script Types**
afterSubmit user event scripts
For more information, see the help topic SuiteScript 2.0 User Event Script Type.

**Governance**
5 units

**Module**
N/redirect Module

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.id  | number | required             | Internal ID of the search. The internal ID is available only when the search is either loaded with `search.load(options)` or after is has been saved with `Search.save()`.
|             |        |                      | Typical values are 55 or 234 or 87, not a value like `customsearch_mysearch`. Any ID prefixed with `customsearch` is a script ID, not the internal system ID for a search. |
redirect.toSearch(options)

Method Description
Method used to redirect a user to an on demand search built in SuiteScript. This method loads a search into the session, and then redirects to a URL that loads the search definition page.

Returns
Void

Supported Script Types
afterSubmit user event scripts
For more information, see the help topic SuiteScript 2.0 User Event Script Type.

Governance
None

Module
N/redirect Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.search</td>
<td>search.Search</td>
<td>required</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/redirect Module Script Sample.

```javascript
var column = ['internalid'];
var filter = [["mainline", "is", "T"]];
var ourNewSearch = search.create(
  id: 'customsearch_test',
  ...)
```
redirect.toSearchResult(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to redirect a user to a search results page. For example, the results from an on demand search created with the N/search Module, or a loaded search that you modified but did not save.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
</tbody>
</table>
| Supported Script Types | afterSubmit user event scripts  
For more information, see the help topic SuiteScript 2.0 User Event Script Type. |
| Governance | None                                                                 |
| Module | N/redirect Module |
| Since | 2015.2 |

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.Search</td>
<td>search.Search</td>
<td>required</td>
<td></td>
</tr>
</tbody>
</table>

redirect.toSuitelet(options)

| Method Description | Method used to redirect the user to a Suitelet.  
For more information about Suitelets, see the help topic SuiteScript 2.0 Suitelet Script Type. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
</tbody>
</table>
| Supported Script Types | Suitelets, beforeLoad user events, and synchronous afterSubmit user events  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance | None                                                                 |
| Module | N/redirect Module |
| Since | 2015.2 |
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.scriptId</td>
<td>string</td>
<td>required</td>
<td>The script ID for the Suitelet.</td>
</tr>
<tr>
<td>options.deploymentId</td>
<td>string</td>
<td>required</td>
<td>The deployment ID for the Suitelet.</td>
</tr>
<tr>
<td>options.isExternal</td>
<td>boolean</td>
<td>optional</td>
<td>The default value is false – indicates an external Suitelet URL.</td>
</tr>
<tr>
<td>options.parameters</td>
<td>Object</td>
<td>optional</td>
<td>Contains additional URL parameters as key/value pairs.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/redirect Module Script Sample.

```javascript
//Add additional code
...
redirect.toSuitelet({
  scriptId: 31,
  deploymentId: 1,
  parameters: {‘custparam_test’:‘helloWorld’}
});
...
//Add additional code
```

`redirect.toTaskLink(options)`

**Method Description**
Method used to redirect a user to a tasklink.

**Returns**
Void

**Supported Script Types**
Suitelets, beforeLoad user events, and synchronous afterSubmit user events

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/redirect Module

**Since**
2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The taskId for a tasklink</td>
</tr>
</tbody>
</table>
### N/redirect Module

For a list of supported task IDs, see the help topic [Task IDs](#).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.parameters</td>
<td>Object</td>
<td>optional</td>
<td>Contains additional URL parameters as key/value pairs.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/redirect Module Script Sample](#).

```javascript
//Add additional code
...
redirect.toTaskLink({
    id: ADMI_SHIPPING,
    parameters: {'custparam_test': 'helloWorld'}
});
...
//Add additional code
```

### N/render Module

The render module encapsulates functionality for printing, PDF creation, form creation from templates, and email creation from templates.

**Note**: Direct manipulation of the print URL is **not** supported.

- N/render Module Members
- EmailMergeResult Object Members
- TemplateRenderer Object Members
- N/render Module Script Sample

#### N/render Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>render._EMAIL_MERGE_RESULT</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates an email merge result</td>
</tr>
<tr>
<td>Object</td>
<td>render.TEMPLATE_RENDERER</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Encapsulates a template engine object that produces HTML and PDF printed forms utilizing advanced PDF/HTML template capabilities</td>
</tr>
<tr>
<td>Method</td>
<td>render.bom(options)</td>
<td>file.File</td>
<td>Server-side scripts</td>
<td>Creates a PDF or HTML file object containing a bill of materials</td>
</tr>
<tr>
<td>Method</td>
<td>render.create()</td>
<td>render._TEMPLATE_RENDERER</td>
<td>Server-side scripts</td>
<td>Creates a render.TemplateRenderer object</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>render.###mergeEmail###(options)</td>
<td>render.###EmailMerge###Result</td>
<td>Server-side scripts</td>
<td>Creates a render.EmailMergeResult object</td>
<td></td>
</tr>
<tr>
<td>render.###packingSlip###(options)</td>
<td>file.File</td>
<td>Server-side scripts</td>
<td>Creates a PDF or HTML file object containing a packing slip</td>
<td></td>
</tr>
<tr>
<td>render.###pickingTicket###(options)</td>
<td>file.File</td>
<td>Server-side scripts</td>
<td>Creates a PDF or HTML file object containing a picking ticket</td>
<td></td>
</tr>
<tr>
<td>render.###statement###(options)</td>
<td>file.File</td>
<td>Server-side scripts</td>
<td>Creates a PDF or HTML file object containing a statement</td>
<td></td>
</tr>
<tr>
<td>render.###transaction###(options)</td>
<td>file.File</td>
<td>Server-side scripts</td>
<td>Creates a PDF or HTML file object containing a transaction</td>
<td></td>
</tr>
<tr>
<td>render.###xmlToPdf###(options)</td>
<td>file.File</td>
<td>Server-side scripts</td>
<td>Passes XML to the BFO tag library (which is stored by NetSuite), and returns a PDF file</td>
<td></td>
</tr>
<tr>
<td>Enum</td>
<td>render.DataSource</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported data source types</td>
</tr>
<tr>
<td></td>
<td>render.PrintMode</td>
<td>enum</td>
<td>Server-side scripts</td>
<td>Holds the string values for supported print output types</td>
</tr>
</tbody>
</table>

**EmailMergeResult Object Members**

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>EmailMergeResult.body</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>The body of the email distribution in string format</td>
</tr>
<tr>
<td></td>
<td>EmailMergeResult.subject</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>The subject of the email distribution in string format</td>
</tr>
</tbody>
</table>

**TemplateRenderer Object Members**

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>TemplateRenderer.###addCustomDataSource###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Adds an XML file or JSON object to an advanced template as a custom data source</td>
</tr>
<tr>
<td></td>
<td>TemplateRenderer.###addRecord###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Binds a record to a template variable</td>
</tr>
<tr>
<td></td>
<td>TemplateRenderer.###addSearchResults###(options)</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Binds a search result to a template variable</td>
</tr>
<tr>
<td></td>
<td>TemplateRenderer.###renderAsPdf###()</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Uses an advanced template to produce a PDF printed form</td>
</tr>
<tr>
<td></td>
<td>TemplateRenderer.###renderPdfToResponse###()</td>
<td>void</td>
<td>Server-side scripts</td>
<td>Renders PDF template content as a server response</td>
</tr>
<tr>
<td></td>
<td>TemplateRenderer.###renderAsString###()</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns template content in string form</td>
</tr>
</tbody>
</table>
### TemplateRenderer

**setTemplateById**(options)

- **Return Type**: `void`
- **Supported Script Types**: Server-side scripts
- **Description**: Sets the template using the internal ID.

**setTemplateByScriptId**(options)

- **Return Type**: `void`
- **Supported Script Types**: Server-side scripts
- **Description**: Sets the template using the script ID.

**renderToResponse**(options)

- **Return Type**: `void`
- **Supported Script Types**: Server-side scripts
- **Description**: Renders HTML template content as a server response.

### Property

**templateContent**

- **Return Type**: `string`
- **Supported Script Types**: Server-side scripts
- **Description**: Content of template.

---

**N/render Module Script Sample**

**Note:** These sample scripts use the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment. For help with finding a record's internal ID, see the help topic How do I find a record's internal ID?

The following example generates a PDF file from a raw XML string.

```javascript
/**
 * @NApiVersion 2.x
 * */
require(['N/render'],
function(render) {
    function generatePdfFileFromRawXml() {
        var xmlStr = '<?xml version="1.0"?>
          <!DOCTYPE pdf PUBLIC "-//big.faceless.org//report" "report-1.1.dtd">
          <pdf>
          <body font-size="18">
          Hello World!
          </body>
          </pdf>"
        var pdfFile = render.xmlToPdf({
          xmlString: xmlStr
        });
        generatePdfFileFromRawXml();
    }
});
```

The following example renders a transaction record into an HTML page.

**Note:** The `entityId` value in this sample is a placeholder. Before using this sample, replace the placeholder values with valid values from your NetSuite account.

```javascript
/**
 * @NApiVersion 2.x
 * */
require(['N/render'],
function(render) {
    function renderTransactionToHtml() {
        var transactionFile = render.transaction({
          entityId: 23,
          printMode: render.PrintMode.HTML
        });
    }
});
```
The following example renders an invoice into a PDF file using an XML template in the file cabinet. This example requires the Advanced PDF/HTML Templates feature.

```javascript
/**
 * @NApiVersion 2.x
 */
// This example shows how to render an invoice into a PDF file using an XML template in the file cabinet.
// Note that this example requires the Advanced PDF/HTML Templates feature.
require(["N/render", 'N/file', 'N/record'],
  function(render, file, record) {
    function renderRecordToPdfWithTemplate() {
      var xmlTemplateFile = file.load('Templates/PDF Templates/invoicePDFTemplate.xml');
      var renderer = render.create();
      renderer.templateContent = xmlTemplateFile.getContents();
      renderer.addRecord('grecord', record.load({
        type: record.Type.INVOICE,
        id: 37
      }));
      var invoicePdf = renderer.renderAsPdf();
    }
    renderRecordToPdfWithTemplate();
  });
```

In the preceding example, the invoicePDFTemplate.xml file was referenced in the File Cabinet. This file is similar to the Standard Invoice PDF/HTML Template found in Customization > Forms > Advanced PDF/HTML Templates.

The following example renders search results into a PDF file.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
// This example shows how to render search results into a PDF file.
// Note that this sample is a Suitelet, so it cannot be run in the debugger.
define(["N/render", 'N/search'],
  function(render, search) {
    function onRequest(options) {
      var request = options.request;
      var response = options.response;

      var xmlStr = "<?xml version="1.0" encoding="UTF-8"?>\n" +
        "<!DOCTYPE pdf PUBLIC "/-//big.faceless.org//report" "report-1.1.dtd">\n" +
        "<pdf lang="ru-RU" xml:lang="ru-RU">\n" +
        "<head>\n" +
        "<link name="russianfont" type="font" subtype="opentype" num="1" +
        "src="NetSuiteFonts/verdana.ttf" +" src-bold="NetSuiteFonts/verdanab.ttf" +
```

Note: This sample script uses the `define` function. You cannot use On Demand Debugging to step through a `define` function. You must use Deployed Debugging to step through this script.
render.EmailMergeResult

Object Description
Encapsulates an email merge result.

Use render.mergeEmail(options) to create and return this object.

For a complete list of this object's properties, see EmailMergeResult Object Members.

Supported Script Types
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/render Module

Since
2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```
//Add additional code
...
var mergeResult = render.mergeEmail({
    templateId: 1234,
    entity: {
        type: 'employee',
    }
});
```
EmailMergeResult.body

Property Description: The body of the email distribution in string format

Type: string (read-only)

Supported Script Types: Server-side scripts

Module: N/render Module

Since: 2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
log.debug({
  title: 'Email Body: ',
  details: mergeResultObj.body
});
...
//Add additional code
```

EmailMergeResult.subject

Property Description: The subject of the email distribution in string format

Type: string (read-only)

Supported Script Types: Server-side scripts

Module: N/render Module

Since: 2015.2
render.TemplateRenderer

Object Description
Encapsulates a template engine object that produces HTML and PDF printed forms utilizing advanced PDF/HTML template capabilities.

The template engine object includes methods that pass in a template as string to be interpreted by FreeMarker, and render interpreted content in your choice of two different formats: as HTML output to an nlobjResponse object, or as XML string that can be passed to render.xmlToPdf(options) to produce a PDF.

This object is available when the Advanced PDF/HTML Templates feature is enabled.

For a complete list of this object’s methods and properties, see TemplateRenderer Object Members.

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/render Module
Since
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```
//Add additional code
//Advanced PDF/HTML Templates feature must be enabled
...
var xmlTplFile = file.load('Templates/PDF Templates/invoicePDFTemplate.xml');
var myFile = render.create();
myFile.templateContent = xmlTplFile.getContents();
myFile.addRecord('record', record.load({
  type: record.Type.INVOICE,
  id: 37
}));
var invoicePdf = myFile.renderAsPdf();
...
```
TemplateRenderer.addCustomDataSource(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds XML or JSON as custom data source to an advanced PDF/HTML template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None                                                                   |
| Module             | N/render Module                                                        |
| Since              | 2016.1                                                                 |

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.alias</td>
<td>string</td>
<td>required</td>
<td>Data source alias</td>
</tr>
<tr>
<td>options.format</td>
<td>render.DataSource</td>
<td>required</td>
<td>Data format</td>
</tr>
<tr>
<td>options.data</td>
<td>Object</td>
<td>Document</td>
<td>string</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var renderer = render.create();

var xmlObj = xml.Parser.fromString(xmlString);
var jsonObj = JSON.parse(jsonString);


renderer.addCustomDataSource({
  format: render.DataSource.XML_DOC,
  alias: 'XML',
  data: xmlObj
});
renderer.addCustomDataSource({
  format: render.DataSource.XML_STRING,
  alias: 'XML_STR',
  data: xmlString
});
```
TemplateRenderer.addRecord(options)

Method Description  Binds a record to a template variable.

Returns  Void

Supported Script Types  Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance  None

Module  N/render Module

Since  2015.2

Parameters

Notes: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.templateName</td>
<td>string</td>
<td>required</td>
<td>Name of the record object variable referred to in the template</td>
</tr>
<tr>
<td>options.record</td>
<td>record.Record object</td>
<td>required</td>
<td>The record to add</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

//Add additional code
...
var myContent = renderer.addRecord({
  templateName: 'record',
  record: record.load({
    type: record.Type.CUSTOMER,
    id: 1234
  });
});
TemplateRenderer.addSearchResults(options)

Method Description Binds a search result to a template variable.

Returns Void

Supported Script Types Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance None

Module N/render Module

Since 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.templateName</td>
<td>string</td>
<td>required</td>
<td>Name of the template</td>
</tr>
<tr>
<td>options.searchResult</td>
<td>search.Result object</td>
<td>required</td>
<td>The search result to add</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var rs = search.create({
    type: search.Type.TRANSACTION,
    columns: ['trandate', 'amount', 'entity'],
    filters: []
}).run();
var results = rs.getRange(0, 1000);
var renderer = render.create();
renderer.templateContent = xmlStr;
renderer.addSearchResults({
    //Add additional code
});
```
TemplateRenderer.renderAsPdf()

**Method Description**
Uses the advanced template to produce a PDF printed form.

**Returns**
file.File

**Supported Script Types**
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/render Module

**Since**
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var invoicePdf = renderer.renderAsPdf();
...
//Add additional code
```

TemplateRenderer.renderPdfToResponse()

**Method Description**
Renders a server response into a PDF file.

For example, you can pass in a response to be rendered as a PDF in a browser, or downloaded by a user.

**Returns**
Void

**Supported Script Types**
Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/render Module

**Since**
2015.2

**Parameters**

⚠️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>response</td>
<td>http.ServerResponse</td>
<td>required</td>
<td>Response that will be written to PDF. For example, the response passed from a Suitelet.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var invoicePdf = renderer.renderPdfToResponse({
    response: myServerResponseObj
});
...  
//Add additional code
```

### TemplateRenderer.renderAsString()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Return template content in string form</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/render Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### TemplateRenderer.renderToResponse(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Writes template content to a server response.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td>Void</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/render Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.response</td>
<td>http.ServerResponse</td>
<td>required</td>
<td>Response to write to</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var invoice = renderer.renderToResponse({
  response: myServerResponseObj
});
...
//Add additional code
```

TemplateRenderer.setTemplateById(options)

- **Method Description:** Sets the template using the internal ID
- **Returns:** Void
- **Supported Script Types:** Server-side scripts
  For more information, see the help topic SuiteScript 2.0 Script Types.
- **Governance:** None
- **Module:** N/render Module
- **Since:** 2016.1

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>number</td>
<td>required</td>
<td>Internal ID of the template</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var renderer = render.create();
renderer.setTemplateById(3);
var xml = renderer.renderAsString();
```
For more information, see the help topics Advanced Templates and Advanced PDF/HTML Templates. To find the template ID, search for PDF Templates or Advanced PDF/HTML Templates in Netsuite. When the list of templates is displayed, hover your cursor on the Edit or Customize link. You can also see the ID in the browser’s URL when you click the link. An example of a Standard PDF template with an ID of 4 is /app/crm/common/merge/pdftemplate.nl?id=4. An example of an Advanced HTML template with an ID of 19 is /app/common/custom/advancedprint/pdftemplate.nl?id=19. IDs from both Standard and Advanced Templates are supported.

**TemplateRenderer.setTemplateByScriptId(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Sets the template using the script ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/render Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Parameters**

- **options.scriptId** (string) required: Script ID of the template

**Syntax**

```javascript
// Add additional code
...
var renderer = render.create();
renderer.setTemplateByScriptId({
  scriptId: "STDTMPLPRICELIST"
});
var xml = renderer.renderAsString();
...
// Add additional code
```

**TemplateRenderer.templateContent**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Content of template</th>
</tr>
</thead>
</table>
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
renderer.templateContent = xmlTemplateFile.getContents();
...
//Add additional code
```

render.bom(options)

Method Description
Use this method to create a PDF or HTML object of a bill of material.

Returns
file.File that contains a PDF or HTML document

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 units

Module
N/render Module

Since
2015.2

Parameters
Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.entityId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the bill of material to print</td>
</tr>
<tr>
<td>options.printMode</td>
<td>string</td>
<td>optional</td>
<td>The print output type. Set using the render.PrintMode enum. By default, uses the company/user preference for print output.</td>
</tr>
<tr>
<td>options.inCustLocale</td>
<td>boolean</td>
<td>optional</td>
<td>Applies when advanced templates are used. Print the document in the customer’s locale.</td>
</tr>
</tbody>
</table>

If basic printing is used, this parameter is ignored and the transaction form is printed in the customer’s locale.
**render.create()**

**Method Description**
Use this method to produce HTML and PDF printed forms with advanced PDF/HTML templates.

Creates `render.TemplateRenderer`.

This object includes methods that pass in a template as string to be interpreted by FreeMarker, and render interpreted content in your choice of two different formats: as HTML output to `http.ServerResponse`, or as XML string that can be passed to `render.xmlToPdf(options)` to produce a PDF.

**Returns**
`render.TemplateRenderer`

**Supported Script Types**
Server-side scripts

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Governance**
None

**Module**
`N/render Module`

**Since**
2015.2

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/render Module Script Sample`.

```javascript
//Add additional code
...
var transactionFile = render.bom({
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
});
...
//Add additional code
```

```javascript
//Add additional code
...
var renderer = render.create();
...
//Add additional code
```
render.mergeEmail(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a render.EmailMergeResult object for a mail merge with an existing scriptable email template</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>render.EmailMergeResult</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server-side scripts  
|                   | For more information, see the help topic SuiteScript 2.0 Script Types.                                 |
| Governance        | None                                                                                                 |
| Module            | N/render Module                                                                                       |
| Since             | 2015.2                                                                                               |

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.templateId</td>
<td>number</td>
<td>required</td>
<td>Internal ID of the template</td>
</tr>
<tr>
<td>options.entity</td>
<td>RecordRef</td>
<td>required</td>
<td>Entity</td>
</tr>
<tr>
<td>options.recipient</td>
<td>RecordRef</td>
<td>required</td>
<td>Recipient</td>
</tr>
<tr>
<td>options.customRecord</td>
<td>RecordRef</td>
<td>required</td>
<td>Custom record</td>
</tr>
<tr>
<td>options.supportCaseId</td>
<td>number</td>
<td>required</td>
<td>Support case ID</td>
</tr>
<tr>
<td>options.transactionId</td>
<td>number</td>
<td>required</td>
<td>Transaction ID</td>
</tr>
</tbody>
</table>

**RecordRef**

You can use a RecordRef to designate the record to perform the mail merge on.

**Note:** The RecordRef object encapsulates the type and ID of a particular record instance.

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RecordRef.id</td>
<td>number</td>
<td>required</td>
<td>Internal ID of the record instance</td>
</tr>
<tr>
<td>RecordRef.type</td>
<td>string</td>
<td>required</td>
<td>The record type ID</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code  
...  
var myMergeResult = render.mergeEmail{}  
```
render.packingSlip(options)

Method Description
Use this method to create a PDF or HTML object of a packing slip.

Returns
file.File that contains a PDF or HTML document

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 units

Module
N/render Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.entityId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the packing slip to print</td>
</tr>
<tr>
<td>options.printMode</td>
<td>string</td>
<td>optional</td>
<td>The print output type. Set using the render.PrintMode enum. By default, uses the company/user preference for print output.</td>
</tr>
<tr>
<td>options.formId</td>
<td>number</td>
<td>optional</td>
<td>The packing slip form number</td>
</tr>
<tr>
<td>options.fulfillmentId</td>
<td>number</td>
<td>optional</td>
<td>Fulfillment ID number</td>
</tr>
<tr>
<td>options.inCustLocale</td>
<td>boolean</td>
<td>optional</td>
<td>Applies when advanced templates are used. Print the document in the customer's locale.</td>
</tr>
</tbody>
</table>
render.pickingTicket(options)

Method Description
Use this method to create a PDF or HTML object of a picking ticket.

Returns
file.File that contains a PDF or HTML document

Supported Script Types
Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 units

Module
N/render Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.entityId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the picking ticket to print</td>
</tr>
<tr>
<td>options.printMode</td>
<td>string</td>
<td>optional</td>
<td>The print output type. Set using the render.PrintMode enum. By default, uses the company/user preference for print output.</td>
</tr>
<tr>
<td>options.formId</td>
<td>number</td>
<td>optional</td>
<td>The packing slip form number</td>
</tr>
<tr>
<td>options.shipgroup</td>
<td>number</td>
<td>optional</td>
<td>Shipping group for the ticket</td>
</tr>
</tbody>
</table>
### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.location</td>
<td>number</td>
<td>optional</td>
<td>Location for the ticket</td>
</tr>
<tr>
<td>options.inCustLocale</td>
<td>boolean</td>
<td>optional</td>
<td>Applies when advanced templates are used. Print the document in the customer's locale. If basic printing is used, this parameter is ignored and the transaction form is printed in the customer's locale.</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see *N/render Module Script Sample*.

```javascript
//Add additional code
...
var transactionFile = render.pickingTicket({
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
});
...
//Add additional code
```

### Method Description

Use this method to create a PDF or HTML object of a statement.

### Returns

file.File that contains a PDF or HTML document

### Supported Script Types

Server-side scripts

For more information, see the help topic *SuiteScript 2.0 Script Types*.

### Governance

10 units

### Module

*N/render Module*

### Since

2015.2

### Parameters

ℹ️ **Note:** The options parameter is a JavaScript object.
### Parameter | Type | Required / Optional | Description
--- | --- | --- | ---
options.formId | number | optional | Internal ID of the form to use to print the statement
options.startDate | Date | optional | Date of the oldest transaction to appear on the statement
options.statementDate | Date | optional | Statement date
options.openTransactions | boolean | true | Include only open transactions
options.inCustLocale | boolean | true | Applies when advanced templates are used. Print the document in the customer's locale.
 |  | false | If basic printing is used, this parameter is ignored and the transaction form is printed in the customer's locale.
options.consolidateStatements | boolean | true | Convert all amount values to the base currency

### Syntax

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/render Module Script Sample](#).

```javascript
//Add additional code
...
var transactionFile = render.statement(
    entityId: 23,
    printMode: render.PrintMode.HTML,
    inCustLocale: true
);
...
//Add additional code
```

**render.transaction(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Use this method to create a PDF or HTML object of a transaction.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>File size is limited to 10MB.</td>
</tr>
<tr>
<td>If the Advanced PDF/HTML Templates feature is enabled, you can associate an advanced template with the custom form saved for a transaction. The advanced template is used to format the printed transaction. For details about this feature, see the help topic <a href="#">Advanced PDF/HTML Templates</a>.</td>
<td></td>
</tr>
</tbody>
</table>

**Returns**

*file.File* that contains a PDF or HTML document

**Supported Script Types**

Server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.entityId</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the transaction to print</td>
</tr>
<tr>
<td>options.printMode</td>
<td>enum</td>
<td>optional</td>
<td>The print output type. Set using the render.PrintMode enum. By default, uses the company/user preference for print output.</td>
</tr>
<tr>
<td>options.formId</td>
<td>number</td>
<td>optional</td>
<td>The transaction form number</td>
</tr>
<tr>
<td>options.inCustLocale</td>
<td>boolean</td>
<td>optional</td>
<td>Applies when advanced templates are used. Print the document in the customer's locale. If basic printing is used, this parameter is ignored and the transaction form is printed in the customer's locale.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var transactionFile = render.transaction({
  entityId: 23,
  printMode: render.PrintMode.HTML,
  inCustLocale: true
});
...
//Add additional code
```

render.xmlToPdf(options)

Method Description: Method used to pass XML to the Big Faceless Organization (BFO) tag library (which is stored by NetSuite), and return a PDF file. BFO version 1.1.63 is supported in NetSuite.

Note: File size cannot exceed 10MB.

Returns: file.File

Supported Script Types: Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance 10 units
Module N/render Module
Since 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.xmlString</td>
<td>xml.Document</td>
<td>string</td>
<td>required</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
var pdfFile = render.xmlToPdf({
  xmlString: xmlStr
});
...
//Add additional code
```

render.DataSource

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported data source types. Use this enum to set the options.format parameter.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
<td></td>
</tr>
</tbody>
</table>

Supported Script Types

Server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module N/render Module

Values

- JSON
render Module

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
renderer.addCustomDataSource({
    format: render.DataSource.JSON,
    alias: 'JSON_STR',
    data: jsonString
});
...
//Add additional code
```

render.PrintMode

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Holds the string values for supported print output types. Use this enum to set the options.printMode parameter.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

Supported Script Types

- Server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/render Module

Values

- DEFAULT
- HTML
- PDF

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/render Module Script Sample.

```javascript
//Add additional code
...
printMode: render.PrintMode.HTML
...
//Add additional code
```
## N/runtime Module

Load the runtime module when you want to access the current runtime settings for the script and script deployment, the user currently executing the script, and user-defined sessions.

- **N/runtime Module Members**
- **Script Object Members**
- **Session Object Members**
- **User Object Members**
- **N/runtime Module Script Sample**

### N/runtime Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>runtime.Script</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the runtime settings of the currently executing script.</td>
</tr>
<tr>
<td></td>
<td>runtime.Session</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the user session for the currently executing script.</td>
</tr>
<tr>
<td></td>
<td>runtime.User</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the properties and preferences for the user of the currently executing script.</td>
</tr>
<tr>
<td>Method</td>
<td>runtime.#getCurrentScript</td>
<td>runtime.Script</td>
<td>Client and server-side scripts</td>
<td>Returns a runtime.Script that represents the currently executing script.</td>
</tr>
<tr>
<td></td>
<td>runtime.#getCurrentSession</td>
<td>runtime.Session</td>
<td>Client and server-side scripts</td>
<td>Returns a runtime.Session that represents the user session for the currently executing script.</td>
</tr>
<tr>
<td></td>
<td>runtime.#getCurrentUser</td>
<td>runtime.User</td>
<td>Client and server-side scripts</td>
<td>Returns a runtime.User that represents the properties and preferences for the user of the currently executing script.</td>
</tr>
<tr>
<td></td>
<td>runtime.#isFeatureInEffect</td>
<td>boolean</td>
<td>Client and server-side scripts</td>
<td>Use this method to determine if a particular feature is enabled in a NetSuite account.</td>
</tr>
<tr>
<td>Property</td>
<td>runtime.accountId</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the account ID for the currently logged-in user.</td>
</tr>
<tr>
<td></td>
<td>runtime.envType</td>
<td>runtime.#Env</td>
<td>Client and server-side scripts</td>
<td>Returns the current environment in which the script is executing.</td>
</tr>
<tr>
<td></td>
<td>runtime.#executionContext</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Returns a runtime.ContextType enumeration that represents what triggered the current script.</td>
</tr>
<tr>
<td></td>
<td>runtime.#processorCount</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>The number of processors available to the current account.</td>
</tr>
<tr>
<td></td>
<td>runtime.queueCount</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>The number of scheduled script queues available to the current account.</td>
</tr>
<tr>
<td></td>
<td>runtime.version</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the version of NetSuite that the method is called in. For example, the runtime.version property in an account running NetSuite 2015.2 is <strong>2015.2</strong>.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>--------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Enum</td>
<td>runtime.ContextType</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Enumeration that holds the context information about what triggered the current script. Returned by the runtime.executionContext property of the N/runtime Module.</td>
</tr>
<tr>
<td></td>
<td>runtime.EnvType</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Enumeration that holds all possible environment types that the current script can execute in.</td>
</tr>
<tr>
<td></td>
<td>runtime.Permission</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Enumeration that holds the user permission level for a specific permission ID. Returned by the User.getPermission(options) method.</td>
</tr>
</tbody>
</table>

**Script Object Members**

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Script.getParameter</td>
<td>number</td>
<td>Date</td>
<td>string</td>
</tr>
<tr>
<td></td>
<td>Script.getRemainingUsage</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns a number value for the usage units remaining for the currently executing script.</td>
</tr>
<tr>
<td>Property</td>
<td>Script.deploymentId</td>
<td>string (read-only)</td>
<td>Server-side scripts</td>
<td>Returns the deployment ID for the script deployment on the currently executing script.</td>
</tr>
<tr>
<td></td>
<td>Script.id</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the script ID for the currently executing script.</td>
</tr>
<tr>
<td></td>
<td>Script.logLevel</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the script logging level for the current script execution. This method is not supported on client scripts.</td>
</tr>
<tr>
<td></td>
<td>Script.percentComplete</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Get or set the percent complete specified for the current scheduled script execution. The return value will appear in the % Complete column on the Scheduled Script Status page.</td>
</tr>
<tr>
<td></td>
<td>Script.bundleIds</td>
<td>Array (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns an Array of bundle IDs for the bundles that include the currently executing script.</td>
</tr>
</tbody>
</table>

**Session Object Members**

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Session.get(options)</td>
<td>string</td>
<td>null</td>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>
### Session.set(options)

**Return Type / Value Type:** `void`

**Supported Script Types:** Client and server-side scripts

**Description:** Sets a key and value for a user-defined session object.

### User Object Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>User.getPermission(options)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Returns a user permission level for the specified permission as a <code>runtime.Permission</code> enumeration.</td>
</tr>
<tr>
<td>Method</td>
<td>User.getPreference(options)</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Returns the value of a NetSuite preference. Currently only General Preferences and Accounting Preferences are exposed in SuiteScript. For more information about these preferences names and IDs, see the help topics General Preferences and Accounting Preferences.</td>
</tr>
<tr>
<td>Property</td>
<td>User.department</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the department for the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.email</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the email address of the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.id</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.location</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the location of the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.name</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the name of the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.role</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Return the internal ID of the role for the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.roleCenter</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the center type, or role center, for the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.roleId</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the custom scriptId of the role for the currently logged-in user.</td>
</tr>
<tr>
<td>Property</td>
<td>User.subsidiary</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>Returns the internal ID of the subsidiary for the currently logged-in user.</td>
</tr>
</tbody>
</table>
N/runtime Module Script Sample

** Note: These samples use the `define` function. The NetSuite Debugger cannot step through a `define` function. If you need to step through your code in the NetSuite Debugger, you must use a `require` function.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

The following Suitelet sample writes user and session information for the currently executing script to the response.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define(['N/runtime'],
    function(runtime) {
        function onRequest(context) {
            var remainingUsage = runtime.getCurrentScript().getRemainingUsage();
            var userRole = runtime.getCurrentUser().role;
            runtime.getCurrentSession().set({
                name: 'scope',
                value: 'global'
            });
            var sessionScope = runtime.getCurrentSession().get({
                name: 'scope'
            });
            log.debug('Remaining Usage:', remainingUsage);
            log.debug('Role:', userRole);
            log.debug('Session Scope:', sessionScope);
            context.response.write('Executing under role: ' + userRole + '. Session scope: ' + sessionScope + '.');
        }
        return {
            onRequest: onRequest
        };
    });
```

The following scheduled script creates sales records during runtime and logs the record creation progress.

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType scheduledscript
 */
define(['N/runtime', 'N/record'],
    function(runtime, record) {
        return {
            execute: function(context) {
                var script = runtime.getCurrentScript();
                for (x=0; x<500; x++) {
                    var rec = record.create({
                        type: record.Type.SALES_ORDER
                    });
            
```
```javascript
script.percentComplete = (x * 100)/500;
log.debug({
  title: 'New Sales Orders',
  details: 'Record creation progress: ' + script.percentComplete + '%'
});
});
});
});

runtime.Script

Object Description
Encapsulates the runtime settings of the currently executing script.
Use runtime.getCurrentScript() to return this object.
For a complete list of this object's methods and properties, see Script Object Members.

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/runtime Module

Since
2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug('Script ID: ' + scriptObj.id);
...
//Add additional code
```

Script.getParameter(options)

Method Description
Returns the value of a script parameter for the currently executing script.

Returns
number | Date | string | Array

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/runtime Module

Since
2015.2
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>The name of the script parameter.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code

var scriptObj = runtime.getCurrentScript();
log.debug("Script parameter of custscript1: " + scriptObj.getParameter({name: 'custscript1'}));

//Add additional code
```

Script.getRemainingUsage()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a number value for the usage units remaining for the currently executing script.</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>None</td>
<td>N/runtime Module</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Governance

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code

var scriptObj = runtime.getCurrentScript();
log.debug("Remaining governance units: " + scriptObj.getRemainingUsage());

//Add additional code
```

Script.deploymentId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the deployment ID for the script deployment on the currently executing script.</td>
<td>string (read-only)</td>
</tr>
</tbody>
</table>
Supported Script Types

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the script ID for the currently executing script.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>

Since 2015.2

Syntax

```javascript
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Deployment Id: " + scriptObj.deploymentId);
...

//Add additional code
```

Script.logLevel

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the script logging level for the current script execution. This method is not supported on client scripts.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns one of the following values:</td>
</tr>
<tr>
<td></td>
<td>■ DEBUG</td>
</tr>
</tbody>
</table>
N/runtime Module

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

#### Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/runtime Module Script Sample](#).

```javascript
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
log.debug("Logging level: " + scriptObj.logLevel);
...
//Add additional code
```

---

### Script.percentComplete

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Get or set the percent complete specified for the current scheduled script execution. The return value appears in the % Complete column on the Scheduled Script Status page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Important: This property throws SSS_OPERATION_UNAVAILABLE if the currently executing script is not a scheduled script.

```javascript
//Gets the percentage of records completed
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
if (scriptObj.executionContext == ContextType.SCHEDULED)
{
```

---
log.debug({
  details: "Script percent complete: " + scriptObj.percentComplete
});
... 
}
...

//Sets the percent complete
...
var script = runtime.getCurrentScript();
for (x=0; x<500; x++) {
  var rec = record.create(
    type:record.Type.SALES_ORDER
  );
  script.percentComplete = (x * 100)/500;
  log.debug(
    title: 'New Sales Orders',
    details: "Record creation progress: " + script.percentComplete + ":%"
  );
}
...
//Add additional code

---

### Script.bundleIds

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns an Array of bundle IDs for the bundles that include the currently executing script.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Array (read-only)</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Client and server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/runtime Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
var bundleArr = scriptObj.bundleIds;
...
//Add additional code
```

---

### Script.apiVersion

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the current script runtime version</th>
</tr>
</thead>
</table>
**Runtime Module**

**Type**

String

**Supported Script Types**

Client and server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/runtime Module

**Since**

2015.2

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var scriptObj = runtime.getCurrentScript().apiVersion;
...
//Add additional code
```

---

**runtime.Session**

**Object Description**

Encapsulates the user session for the currently executing script. Use this object to set and get user-defined objects for the current user session. Use the objects to track user-related session data. For example, you can gather information about the user scope, budget, or business problems.

Use `Session.set(options)` to set session object values and then use `Session.get(options)` to retrieve the values.

For a complete list of this object's methods, see Session Object Members.

**Supported Script Types**

Client and server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/runtime Module

**Since**

2015.2

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({name: "myKey", value: "myValue"});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"});
...
//Add additional code
```
Session.get(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns the user-defined session object value associated with the session object key.</th>
<th>If the key does not exist, this method returns null.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
<td>null</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>String used as a key to store the runtime.Session.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see link back to N/runtime Module Script Sample.

```
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({name: "myKey", value: "myValue"});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"});
...
//Add additional code
```

Session.set(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Sets a key and value for a user-defined runtime.Session. Use Session.get(options) to retrieve the object value after you set it.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Key used to store the <code>runtime.Session</code>.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>Required</td>
<td>Value to associate with the key in the user session.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see link back to [N/runtime Module Script Sample](#).

```javascript
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
sessionObj.set({
    name: "myKey",
    value: "myValue"
});
log.debug("Session object myKey value: " + sessionObj.get({name: "myKey"});
...
//Add additional code
```

## runtime.User

**Object Description**
Encapsulates the properties and preferences for the user of the currently executing script.

For a complete list of this object's methods and properties, see [User Object Members](#).

**Supported Script Types**
Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/runtime Module

**Since**
2015.2

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/runtime Module Script Sample](#).

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
```
User.getPermission(options)

**Method Description**

Returns a user permission level for the specified permission as a `runtime.Permission` enumeration.

**Returns**

string

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

None

**Module**

N/runtime Module

**Since**

2015.2

### Parameters

*Note:* The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Internal ID of a permission. For a list of permission IDs, see <a href="#">Permission Names and IDs</a>.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see link back to [N/runtime Module Script Sample](#).

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("User permission of ADMI_ACCOUNTING:" +
    (userObj.getPermission('ADMI_ACCOUNTING') ==
    runtime.Permission.FULL?'FULL':userObj.getPermission('ADMI_ACCOUNTING')));
...
//Add additional code
```

User.getPreference(options)

**Method Description**

Returns the value of a NetSuite preference.

Currently only [General Preferences](#) and [Accounting Preferences](#) are exposed in SuiteScript. For more information about these preferences names and IDs, see the help topics [General Preferences](#) and [Accounting Preferences](#).

You can also view General Preferences by going to Setup > Company > General Preferences. View Accounting Preferences by going to Setup > Accounting > Accounting Preferences.
If you want to change the value of a General or Accounting preference using SuiteScript 2.0, you must load each preference page using `config.load(options)`, where `options.name` is either `COMPANY_PREFERENCES` or `ACCOUNTING_PREFERENCES`. The `config.load(options)` method returns a `record.Record`. You can use the `Record.setValue(options)` method to set the preference.

**Note:** The permission level will be Permission.FULL if the script is configured to execute as admin. You can configure a script to execute as admin by selecting “administrator” from the Execute as Role field on Script Deployment page.

<table>
<thead>
<tr>
<th>Returns</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Internal ID of the preference. For a list of preference IDs, see Preference Names and IDs.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see link back to N/runtime Module Script Sample.

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("User preference for emailemployeeonapproval: " + userObj.getPreference({name: "emailemployeeonapproval"}));
...
//Add additional code
```

**User.department**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the internal ID of the department for the currently logged-in user.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
**User.email**

**Property Description**

Returns the email address of the currently logged-in user.

To use this property, the email field on the user employee record must contain an email address.

**Note:** In a shopping context where the shopper is recognized but not logged in, this method can be used to return the shopper’s email, instead of getting it from the customer record.

**Type**

string (read-only)

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/runtime Module

**Since**

2015.2

---

**User.id**

**Property Description**

Returns the internal ID of the currently logged-in user.

**Type**

number (read-only)

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).
Module: N/runtime Module

Since: 2015.2

Syntax

**User.location**

**Property Description**: Returns the internal ID of the location of the currently logged-in user.

**Type**: number (read-only)

**Supported Script Types**: Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**: N/runtime Module

**Since**: 2015.2

Syntax

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user: "+ userObj.id);
...
//Add additional code
```

**User.name**

**Property Description**: Returns the name of the currently logged-in user.

**Type**: string (read-only)

**Supported Script Types**: Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**: N/runtime Module

**Since**: 2015.2

Syntax

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user location: "+ userObj.location);
...
//Add additional code
```
**User.role**

**Property Description**
Return the internal ID of the role for the currently logged-in user.

**Type**
number (read-only)

**Supported Script Types**
Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/runtime Module

**Since**
2015.2

**Syntax**

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Name of current user: " + userObj.name);
...
//Add additional code
```

**User.roleCenter**

**Property Description**
Returns the string value of the center type, or role center, for the currently logged-in user.

The NetSuite user interface adjusts automatically to different users' business needs. For each user, NetSuite displays a variable set of tabbed pages, called a center, based on the user's assigned role. Each NetSuite center provides, for users with related roles, the pages and links they need to do their jobs.

For more information about NetSuite centers, see the help topic [Centers Overview](#).

**Type**
string (read-only)

**Supported Script Types**
Client and server-side scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).
Module | N/runtime Module
---|---
Since | 2015.2

**Syntax**

**User.roleId**

| Property Description | Returns the custom scriptId of the role for the currently logged-in user. You can use this value instead of the internal ID for the role. When bundling a custom role, the internal ID number of the role in the target account can change after the bundle is installed. Therefore, in the target account you can use this property to return the unique/custom scriptId assigned to the role.
| Type | string
| Supported Script Types | Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**User.subsidiary**

| Property Description | Returns the internal ID of the subsidiary for the currently logged-in user.
| Type | number (read-only)
| Supported Script Types | Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Module</th>
<th>N/runtime Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
log.debug("Internal ID of current user subsidiary: "+userObj.subsidiary);
...
//Add additional code
```

**runtime.getCurrentScript()**

**Method Description**

Returns a `runtime.Script` that represents the currently executing script.

Use this method to get properties and parameters of the currently executing script and script deployment. If you want to get properties for the session or user, use `runtime.getCurrentSession()` or `runtime.getCurrentUser()` instead.

**Returns**

`runtime.Script`

**Supported Script Types**

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/runtime Module

**Since**

2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var scriptObj = runtime.getCurrentScript();
...
//Add additional code
```

**runtime.getCurrentSession()**

**Method Description**

Returns a `runtime.Session` that represents the user session for the currently executing script.

`runtime.Session`
Use this method to get session objects for the current user session. If you want to get properties for the script or user, use runtime.getCurrentScript() or runtime.getCurrentUser() instead.

Returns

runtime.Session

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/runtime Module

Since

2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var sessionObj = runtime.getCurrentSession();
...
//Add additional code
```

runtime.getCurrentUser()

Method Description

Returns a runtime.User that represents the properties and preferences for the user of the currently executing script. Use this method to get session objects for the current user session. If you want to get properties for the script or session, use runtime.getCurrentScript() or runtime.getCurrentSession() instead.

Returns

runtime.User

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/runtime Module

Since

2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
var userObj = runtime.getCurrentUser();
...
```

SuiteScript 2.0 API Reference
runtime.isFeatureInEffect(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Use this method to determine if a particular feature is enabled in a NetSuite account. These are the features that appear on the Enable Features page at Setup &gt; Company &gt; Enable Features.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>boolean true</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.feature</td>
<td>string</td>
<td>Required</td>
<td>The internal ID of the feature to check. For a list of feature internal IDs, see the help topic Feature Names and IDs.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
log.debug('Advanced Billing feature is enabled: ' + runtime.isFeatureInEffect({feature: "ADVBILLING"}));
...
//Add additional code
```
runtime.envType

Property Description
Returns the current environment in which the script is executing.
This property returns one of the values from the runtime.EnvType enumeration.

Type
runtime.EnvType

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/runtime Module

Since
2015.2

runtime.executionContext

Property Description
Property that describes what triggered the current script. This value is set by the runtime.ContextType enumeration.

Type
runtime.ContextType

Supported Script Types
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/runtime Module

Since
2015.2
runtime.processorCount

The number of processors available to the currently logged in account. SuiteCloud Processors is the current system used to execute (process) scheduled scripts and map/reduce scripts. This property is helpful if you are a SuiteApp developer and your script needs to know the total number of processors available to a deployment.

For scheduled script deployments that continue to use queues, use runtime.queueCount. With the introduction of SuiteCloud Processors, map/reduce script deployments and new scheduled script deployments no longer use queues, but pre-existing scheduled script deployments continue to use queues until the queues are removed (see the help topic SuiteCloud Processors – Supported Task Types).

Be aware that the number of processors available may not be the same as the number of queues available. For more information, see the help topic SuiteCloud Plus Settings.

The runtime.processorCount property reflects the number of processors available to an account. It is not impacted by changes to deployments. The value is the same regardless of whether deployments continue to use queues. For more information, see the help topic SuiteCloud Processors – Supported Task Types.

For more information on scheduled scripts, see the help topic SuiteScript 2.0 Scheduled Script Type. For more information on map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Type.

Type: number (read-only)

Module: N/runtime Module

Since: 2018.1

Syntax

```
if (runtime.executionContext === runtime.ContextType.USEREVENT)
    return;
```

```
log.debug("Number of processors available: " + runtime.processorCount);
```

```
//Add additional code
```
**runtime.queueCount**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The number of queues available to the currently logged in account.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>SuiteCloud Processors</strong> is the current system used to execute (process) scheduled scripts and map/reduce scripts. This property is helpful if you are a SuiteApp developer and your script needs to know the total number of queues available to a deployment.</td>
</tr>
<tr>
<td></td>
<td>For map/reduce script deployments, use <strong>runtime.processorCount</strong>. With the introduction of SuiteCloud Processors, no map/reduce script deployments use queues (see the help topic <strong>SuiteCloud Processors – Supported Task Types</strong>).</td>
</tr>
<tr>
<td></td>
<td>Be aware that the number of queues available may not be the same as the number of processors available (see the help topic <strong>SuiteCloud Plus Settings</strong>).</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> If all scheduled script deployments in an account are configured to no longer use queues (see the help topic <strong>SuiteCloud Processors – Supported Task Types</strong>), the value of <strong>runtime.queueCount</strong> is unchanged. This property reflects the number of queues available to an account. It is not impacted by changes to deployments.</td>
</tr>
</tbody>
</table>

For more information on scheduled scripts, see the help topic **SuiteScript 2.0 Scheduled Script Type**.

<table>
<thead>
<tr>
<th>Type</th>
<th>number (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <strong>SuiteScript 2.0 Script Types</strong>.</td>
</tr>
<tr>
<td>Module</td>
<td>N/runtime Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
log.debug("Number of queues available: "+runtime.queueCount);
...
//Add additional code
```

**runtime.version**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the version of NetSuite that the method is called in. For example, the runtime.version property in an account running NetSuite 2015.2 is <strong>2015.2</strong>.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Use this method, for example, when installing a bundle in another NetSuite accounts and you want to know the version number before installing the bundle.</td>
</tr>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>
N/runtime Module

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/runtime Module Script Sample.

```javascript
//Add additional code
...
log.debug("Current NetSuite version: " + runtime.version);
...
//Add additional code
```

runtime.ContextType

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration used to set the runtime.executionContext property. The runtime.executionContext property describes what triggered the current script.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

Supported Script Types

Client and server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/runtime Module

Since

2015.2

Values

<table>
<thead>
<tr>
<th>Enum Value</th>
<th>Sets runtime ExecutionContext Property To</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACTION</td>
<td>ACTION</td>
</tr>
<tr>
<td>BUNDLE_INSTALLATION</td>
<td>BUNDLEINSTALLATION</td>
</tr>
<tr>
<td>CLIENT</td>
<td>CLIENT</td>
</tr>
<tr>
<td>CONSOLRATEADJUSTOR</td>
<td>CONSOLRATEADJUSTOR</td>
</tr>
<tr>
<td>CSV_IMPORT</td>
<td>CSVIMPORT</td>
</tr>
<tr>
<td>CUSTOMGLLINES</td>
<td>CUSTOMGLLINES</td>
</tr>
<tr>
<td>CUSTOM_MASSUPDATE</td>
<td>CUSTOMMSSUPDATE</td>
</tr>
<tr>
<td>DEBUGGER</td>
<td>DEBUGGER</td>
</tr>
<tr>
<td>EMAIL_CAPTURE</td>
<td>EMAILCAPTURE</td>
</tr>
<tr>
<td>Enum Value</td>
<td>Sets runtime.ExecutionContext Property To</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>MAP_REDUCE</td>
<td>MAPREDUCE</td>
</tr>
<tr>
<td>NONE</td>
<td>NONE</td>
</tr>
<tr>
<td>PAYMENTGATEWAY</td>
<td>PAYMENTGATEWAY</td>
</tr>
<tr>
<td>PORTLET</td>
<td>PORTLET</td>
</tr>
<tr>
<td>PROMOTIONS</td>
<td>PROMOTIONS</td>
</tr>
<tr>
<td>RESTLET</td>
<td>RESTLET</td>
</tr>
<tr>
<td>SCHEDULED</td>
<td>SCHEDULED</td>
</tr>
<tr>
<td>SHIPPING_PARTNERS</td>
<td>SHIPPINGPARTNERS</td>
</tr>
<tr>
<td>SUITELET</td>
<td>SUITELET</td>
</tr>
<tr>
<td>TAX_CALCULATION</td>
<td>TAXCALCULATION</td>
</tr>
<tr>
<td>USEREVENT</td>
<td>USEREVENT</td>
</tr>
<tr>
<td>USER_INTERFACE</td>
<td>USERINTERFACE</td>
</tr>
<tr>
<td>WEBAPPLICATION</td>
<td>WEBAPPLICATION</td>
</tr>
<tr>
<td>WEBSERVICES</td>
<td>WEBSERVICES</td>
</tr>
<tr>
<td>WEBSTORE</td>
<td>WEBSTORE</td>
</tr>
<tr>
<td>WORKFLOW</td>
<td>WORKFLOW</td>
</tr>
</tbody>
</table>

**runtime.EnvType**

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds all possible environment types that the current script can execute in. One of these values is returned by the runtime.envType property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/runtime Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Values**

- SANDBOX
- PRODUCTION
runtime.Permission

**Enum Description**
Enumeration that holds the user permission level for a specific permission ID. Returned by the `User.getPermission(options)` method. See the help topic Permission Names and IDs. For information on working with NetSuite permissions, see the help topic NetSuite Permissions Overview.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/runtime Module

**Since**
2015.2

**Values**
- FULL
- EDIT
- CREATE
- VIEW
- NONE

N/search Module

Load the search module to create and run on demand or saved searches and analyze and iterate through the search results. You can search for a single record by keywords, create saved searches, search for duplicate records, or return a set of records that match filters you define.

You also have the option to paginate search results and construct navigation that jumps between the next and previous pages. Due to the performance benefits, this is a suitable approach for working with a large result set.

- N/search Module Members
- Search Object Members
- Result Object Members
- Column Object Members
- Filter Object Members
- Page Object Members
- PagedData Object Members
- PageRange Object Members
### ResultSet Object Members

### N/search Module Script Samples

#### N/search Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>search.Search</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a NetSuite search. Use the methods available to the Search object to create a search, run a search, or save a search.</td>
</tr>
<tr>
<td>search.Result</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulate a single search result row. Use the methods and properties for the Result object to get the column values for the result row.</td>
<td></td>
</tr>
<tr>
<td>search.Column</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a single search column in a search.Search object. Use the methods and properties available to the Column object to get or set Column properties.</td>
<td></td>
</tr>
<tr>
<td>search.Filter</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a search filter used in a search. Use the properties for the Filter object to get and set the filter properties.</td>
<td></td>
</tr>
<tr>
<td>search.ResultSet</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a set of search results returned by Search.run().</td>
<td></td>
</tr>
<tr>
<td>search.Page</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a set of search results for a single search page.</td>
<td></td>
</tr>
<tr>
<td>search.PagedData</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Holds metadata about a paginated query.</td>
<td></td>
</tr>
<tr>
<td>search.PageRange</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Defines the page range to bound the result set for a paginated query.</td>
<td></td>
</tr>
<tr>
<td>search.Setting</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a search setting. Search settings let you specify search parameters that are typically available only in the UI.</td>
<td></td>
</tr>
</tbody>
</table>

### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search.##create(options)</td>
<td>Creates a new search and returns it as a search.Search object.</td>
</tr>
<tr>
<td>search.##create.##promise(options)</td>
<td>Creates a new search asynchronously and returns it as a search.Search object.</td>
</tr>
<tr>
<td>search.load(options)</td>
<td>Loads an existing saved search and returns it as a search.Search object.</td>
</tr>
<tr>
<td>search.load.##promise(options)</td>
<td>Loads an existing saved search asynchronously and returns it as a search.Search object.</td>
</tr>
<tr>
<td>search.##delete(options)</td>
<td>Deletes an existing saved search asynchronously and returns it as a search.Search object.</td>
</tr>
<tr>
<td>search.##delete.##promise(options)</td>
<td>Deletes an existing saved search and returns it as a search.Search object.</td>
</tr>
<tr>
<td>search.##duplicates(options)</td>
<td>Performs a search for duplicate records based on the duplicate detection configuration for the account. Returns an array of search.Result objects.</td>
</tr>
</tbody>
</table>
## Search Object Members

The following members are called on `search.Search`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>Search.save()</code></td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Saves a search created by <code>search.create(options)</code> or loaded with <code>search.load(options)</code>. Returns the internal ID of the saved search.</td>
</tr>
<tr>
<td></td>
<td><code>Search.save.promise()</code></td>
<td>number</td>
<td>Client scripts</td>
<td>Asynchronously saves a search created by <code>search.create(options)</code> or loaded</td>
</tr>
</tbody>
</table>
### Member Type

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Search.run()</code></td>
<td><code>search.ResultSet</code></td>
<td>Client and server-side scripts</td>
<td>Runs an on demand search created with <code>search.create(options)</code> or a search loaded with <code>search.load(options)</code>, returning the results as a <code>search.ResultSet</code>.</td>
</tr>
<tr>
<td><code>Search.runPaged()</code></td>
<td><code>search.PagedData</code></td>
<td>Client and server-side scripts</td>
<td>Runs the current search and returns a <code>search.PagedData</code> Object.</td>
</tr>
<tr>
<td><code>Search.runPaged.promise()</code></td>
<td><code>search.PagedData</code></td>
<td>Client scripts</td>
<td>Asynchronously runs the current search and returns a <code>search.PagedData</code> Object.</td>
</tr>
</tbody>
</table>

### Property

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Search.searchType</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Search type on which a search is based.</td>
</tr>
<tr>
<td><code>Search.searchId</code></td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Internal ID of a search.</td>
</tr>
<tr>
<td><code>Search.filters</code></td>
<td><code>search.Filter[]</code></td>
<td>Client and server-side scripts</td>
<td>Filters for the search as an array of <code>search.Filter</code> objects.</td>
</tr>
<tr>
<td><code>Search.filterExpression</code></td>
<td>Object[]</td>
<td>Client and server-side scripts</td>
<td>Search filter expression for the search as an array of expression objects.</td>
</tr>
<tr>
<td><code>Search.columns</code></td>
<td><code>search.Column[]</code></td>
<td>Client and server-side scripts</td>
<td>Columns to return for this search as an array of <code>search.Column</code> objects or a string array of column names.</td>
</tr>
<tr>
<td><code>Search.settings</code></td>
<td><code>search.Setting[]</code></td>
<td>Client and server-side scripts</td>
<td>Search settings for this search as an array of <code>search.Setting</code> objects or a string array of column names.</td>
</tr>
<tr>
<td><code>Search.title</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Title for a saved search. Use this property to set the title for a search before you save it for the first time.</td>
</tr>
<tr>
<td><code>Search.id</code></td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Script ID for a saved search, starting with <code>customsearch</code>.</td>
</tr>
<tr>
<td><code>Search.isPublic</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Value is <code>true</code> if the search is public, or <code>false</code> if it is not.</td>
</tr>
</tbody>
</table>

### Column Object Members

The following members are called on `search.Column`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Column.setWhenOrderedBy()</code></td>
<td><code>search.Column</code></td>
<td>Client and server-side scripts</td>
<td>Returns the search column for which the minimal or maximal value should be found when returning the <code>search.Column</code> value.</td>
</tr>
<tr>
<td><code>Column.name</code></td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Name of a search column as a string.</td>
</tr>
<tr>
<td><code>Column.join</code></td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Join ID for a search column as a string.</td>
</tr>
</tbody>
</table>
### Filter Object Members

The following members are called on `search.Filter`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Filter.name</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Name or internal ID of the search field.</td>
</tr>
<tr>
<td></td>
<td>Filter.join</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Join ID for the search filter.</td>
</tr>
<tr>
<td></td>
<td>Filter.operator</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Operator used for the search filter.</td>
</tr>
<tr>
<td></td>
<td>Filter.summary</td>
<td>search.Summary</td>
<td>Client and server-side scripts</td>
<td>Summary type for the search filter.</td>
</tr>
<tr>
<td></td>
<td>Filter.formula</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Formula used by the search filter.</td>
</tr>
</tbody>
</table>

### Page Object Members

The following members are called on the `search.Page`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Page.next()</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Gets the next segment of data from a paginated search.</td>
</tr>
<tr>
<td></td>
<td>Page.next.promise()</td>
<td>void</td>
<td>Client scripts</td>
<td>Asynchronously gets the next segment of data from a paginated search</td>
</tr>
<tr>
<td></td>
<td>Page.prev()</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Gets the previous segment of data from a paginated search</td>
</tr>
<tr>
<td></td>
<td>Page.prev.promise()</td>
<td>void</td>
<td>Client scripts</td>
<td>Asynchronously gets the previous segment of data from a paginated search</td>
</tr>
</tbody>
</table>
## Page Object Members

The following members are called on `search.Result`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Page.data</td>
<td><code>search.Result[]</code></td>
<td>Client and server-side scripts</td>
<td>The results from a paginated search.</td>
</tr>
<tr>
<td></td>
<td>Page.isFirst</td>
<td>read-only boolean</td>
<td>Client and server-side scripts</td>
<td>Indicates whether a page is the first page of data for a result set.</td>
</tr>
<tr>
<td></td>
<td>Page.isLast</td>
<td>read-only boolean</td>
<td>Client and server-side scripts</td>
<td>Indicates whether a page is the last page of data for a result set.</td>
</tr>
<tr>
<td></td>
<td>Page.pagedData</td>
<td><code>search.PagedData</code></td>
<td>Client and server-side scripts</td>
<td>The <code>PagedData</code> Object used to fetch this <code>Page</code> Object.</td>
</tr>
<tr>
<td></td>
<td>Page.pageRange</td>
<td><code>search.PageRange</code></td>
<td>Client and server-side scripts</td>
<td>The <code>PageRange</code> Object used to fetch this <code>Page</code> Object.</td>
</tr>
</tbody>
</table>

## PagedData Object Members

The following members are called on `search.PagedData`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>PagedData.#fetch(options)</td>
<td><code>search.Page</code></td>
<td>Client and server-side scripts</td>
<td>Retrieves the data within the specified page range.</td>
</tr>
<tr>
<td></td>
<td>PagedData.#fetch.#promise(options)</td>
<td><code>search.Page</code></td>
<td>Client scripts</td>
<td>Asynchronously retrieves the data within the specified page range.</td>
</tr>
<tr>
<td>Property</td>
<td>PagedData.count</td>
<td>read-only number</td>
<td>Client and server-side scripts</td>
<td>The total number of results when <code>Search.runPaged(options)</code> was executed.</td>
</tr>
<tr>
<td></td>
<td>PagedData.pageRanges</td>
<td><code>search.PageRange[]</code></td>
<td>Client and server-side scripts</td>
<td>The collection of <code>PageRange</code> objects that divide the entire result set into smaller groups.</td>
</tr>
<tr>
<td></td>
<td>PagedData.pageSize</td>
<td>read-only number</td>
<td>Client and server-side scripts</td>
<td>The maximum number of entries per page.</td>
</tr>
<tr>
<td></td>
<td>PagedData.searchDefinition</td>
<td><code>search.Search</code></td>
<td>Client and server-side scripts</td>
<td>The search criteria used when <code>Search.runPaged(options)</code> was executed.</td>
</tr>
</tbody>
</table>

## PageRange Object Members

The following members are called on `search.PageRange`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>PageRange.compoundLabel</td>
<td>read-only string</td>
<td>Client and server-side scripts</td>
<td>Human-readable label with beginning and ending range identifiers.</td>
</tr>
<tr>
<td></td>
<td>PageRange.index</td>
<td>read-only number</td>
<td>Client and server-side scripts</td>
<td>The index of this page range.</td>
</tr>
</tbody>
</table>
Result Object Members

The following members are called on search.Result.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Result.getValue</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Used on formula fields and non-formula (standard) fields to get the value of a specified search return column.</td>
</tr>
<tr>
<td></td>
<td>(options)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result.getValue</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Used on formula and non-formula (standard) fields. Returns the string value of a specified search result column. For convenience, this method takes a single search.Column Object.</td>
</tr>
<tr>
<td></td>
<td>(column)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result.getText</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The text value for a search.Column if it is a stored select field.</td>
</tr>
<tr>
<td></td>
<td>(column)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Result.getText</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The UI display name, or text value, for a search result column. This method is supported only for non-stored select, image, and document fields.</td>
</tr>
<tr>
<td></td>
<td>(options)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Result.recordType</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>The type of record returned in a search result row.</td>
</tr>
<tr>
<td></td>
<td>Result.id</td>
<td>number (read-only)</td>
<td>Client and server-side scripts</td>
<td>The internal ID for the record returned in a search result row.</td>
</tr>
<tr>
<td></td>
<td>Result.columns</td>
<td>search.Column[]</td>
<td>Client and server-side scripts</td>
<td>Array of search.Column objects that encapsulate the columns returned in the search result row.</td>
</tr>
</tbody>
</table>

ResultSet Object Members

The following members are called on search.ResultSet.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>ResultSet.getRange</td>
<td>search.Result[]</td>
<td>Client and server-side scripts</td>
<td>Retrieve a slice of the search result as an array of search.Result objects.</td>
</tr>
<tr>
<td></td>
<td>(options)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ResultSet.getRange</td>
<td>search.Result[]</td>
<td>Client scripts</td>
<td>Asynchronously retrieve a slice of the search result as an array of search.Result objects.</td>
</tr>
<tr>
<td></td>
<td>(options)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ResultSet.each</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Use a developer-defined function to invoke on each row in the search results, up to 4000 results at a time.</td>
</tr>
</tbody>
</table>
N/search Module

SuiteScript 2.0 API Reference

Member Type | Name | Return Type / Value Type | Supported Script Types | Description
--- | --- | --- | --- | ---
**ResultSet.**each.**void** | resultSetPromise | Client scripts | Asynchronously use a developer-defined function to invoke on each row in the search results, up to 4000 results at a time.

Property | **ResultSet.columns** | search.Column[] | Client and server-side scripts | An array of search.Column objects that represent the columns returned in the search results.

Setting Object Members

The following members are called on search.Setting.

Member Type | Name | Return Type / Value Type | Supported Script Types | Description
--- | --- | --- | --- | ---
Property | **Setting.name** | read-only string | Client and server-side scripts | The name of the search parameter.

**Setting.value** | read-only string | Client and server-side scripts | The value of the search parameter.

N/search Module Script Samples

In your NetSuite account, the One World feature needs to be enabled in the account for the samples to work. These samples are designed to run from a OneWorld account.

---

### Note:

These sample scripts use the require function so that you can copy it into the debugger and test it. Keep in mind that you must use the define function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

---

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

The following examples create a saved search on the sales order record.

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

// This example creates a saved search on the salesorder record
require(['N/search'],

function(search) {
    function createSearch() {
        var mySalesOrderSearch = search.create({
            type: search.Type.SALES_ORDER,
            title: 'My SalesOrder Search',
            id: 'customsearch_my_so_search',
            columns: ['entity', 'subsidiary', 'name', 'currency'],
            filters: [
                ['mainline', 'is', 'T'],
                'and', ['subsidiary.name', 'contains', 'CAD']
            ]
        });
    }
})
```
The following example loads and runs a search on the sales order record, and uses a callback function on the results.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/search'],
    function(search) {
        function loadAndRunSearch() {
            var mySearch = search.load({
                id: 'customsearch_my_so_search'
            });
            mySearch.run().each(function(result) {
                var entity = result.getValue({
                    name: 'entity'
                });
                var subsidiary = result.getValue({
                    name: 'subsidiary'
                });
            });
        }
        loadAndRunSearch();
    });
```
The following example loads and runs a search on the sales order record, and gets the first 100 rows of results.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/search'],
function(search) {
  function runSearchAndFetchResult() {
    var mySearch = search.load({
      id: 'customsearch_my_so_search'
    });
    var searchResult = mySearch.run().getRange({
      start: 0,
      end: 100
    });
    for (var i = 0; i < searchResult.length; i++) {
      var entity = searchResult[i].getValue({
        name: 'entity'
      });
      var subsidiary = searchResult[i].getValue({
        name: 'subsidiary'
      });
    }
  }
  runSearchAndFetchResult();
});
```

The following example loads and runs a search on the sales order record, and uses a callback function on the paginated results.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/search'],
function(search) {
  function loadAndRunSearch() {
    var mySearch = search.load({
      id: 'customsearch_my_so_search'
    });
    var myPagedData = mySearch.runPaged();
    myPagedData.pageRanges.forEach(function(pageRange){
      var myPage = myPagedData.fetch({index: pageRange.index});
      myPage.data.forEach(function(result){
        var entity = result.getValue({
          name: 'entity'
        });
        var subsidiary = result.getValue({
          name: 'subsidiary'
        });
      });
    });
  }
  loadAndRunSearch();
});
```
The following example deletes a saved search.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/search'],
    function(search) {
        function deleteSearch() {
            search.delete({
                id: 'customsearch_my_so_search'
            });
        }
        deleteSearch();
    });
```

**search.Search**

**Object Description**
Encapsulates a NetSuite search. Use the methods available to search.Search to create a search, run a search, or save a search.

**Note:** You do not need to save the search to run it.

For more information about executing NetSuite searches using SuiteScript, see Searching Overview.

For a complete list of this object's methods and properties, see Search Object Members.

**Supported Script Types**
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/search Module

**Since**
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
...
//Add additional code
```
Search.run()

**Method Description**
Runs an on-demand search created with `search.create(options)` or a search loaded with `search.load(options)`, returning the results as a `search.ResultSet`. Calling this method does not save the search.

Use this method with `search.create(options)` to create and run on-demand searches that are never saved to the database.

After you run a search, you can use `ResultSet.each(callback)` to iterate through the result set and process each result.

**Important:** When you call this method, consider the following:

- Search result sets are not cached. If records applicable to your search are created, modified, or deleted at the same time you are traversing your result set, your result set may change.
- For better performance, consider creating a saved search in the UI and loading it in your script using `search.load(options)` instead of creating the search directly in your script using `search.create(options)`.

**Returns**
`search.ResultSet`

**Governance**
None

**Module**
N/search Module

**Since**
2015.2

**Syntax**

```javascript
// Add additional code
...
function loadAndRunSearch() {
    var mySearch = search.load({
        id: 'customsearch_my_so_search'
    });
    mySearch.run().each(function(result) {
        var entity = result.getValue({
            name: 'entity'
        });
        var subsidiary = result.getValue({
            name: 'subsidiary'
        });
        return true;
    });
    return true;
}
```

Search.runPaged(options)

**Method Description**
Runs the current search and returns summary information about paginated results.

Calling this method does not give you the result set or save the search.
To retrieve data, use `PagedData.fetch(options)`.

### Important
When you use this method to run a paged search, consider the following:

- Search result sets are not cached. If records applicable to your search are created, modified, or deleted at the same time you are traversing your result set, your result set may change.
- This method can return a maximum of 1000 pages of search results.

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.PagedData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.pageSize</td>
<td>number</td>
<td>optional</td>
<td>Maximum number of entries per page</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>There is an upper limit, a lower limit, and a default setting:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- The maximum number allowed is 1000.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- The minimum number allowed is 5.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>- By default, the page size is set to 50 entries per page.</td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see **N/search Module Script Samples**.

```javascript
//Add additional code
...
var mySearch = search.create({
    type: search.Type.CUSTOMER
});

// Run the paged search
var pagedData = mySearch.runPaged({
    pageSize: 50
});
...
// Use the count property to count the
// search results easily
```
var resultCount = mySearch.runPaged({
    pageSize: 50
}).count;
...
//Add additional code

### Search.runPaged.promise(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Runs the current search asynchronously and returns a <code>search.PagedData</code> Object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>For more information about using this method, see <code>Search.runPaged(options)</code>. For additional information on promises, see Promise Object.</td>
</tr>
<tr>
<td>Returns</td>
<td><code>search.PagedData</code></td>
</tr>
<tr>
<td>Synchronous Version</td>
<td><code>Search.runPaged(options)</code></td>
</tr>
</tbody>
</table>
| Supported Script Types | All client-side scripts  
                          Note: For more information, see the help topic SuiteScript 2.0 Client Script Type. |
| Governance         | 5 units                                                                     |
| Module             | `N/search Module`                                                          |
| Since              | 2016.1                                                                     |

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
mySearch.runPaged.promise().then(getPageRangesPromiseChain);
...
//Add additional code
```

### Search.save()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Saves a search created by <code>search.create(options)</code> or loaded with <code>search.load(options)</code>. Returns the internal ID of the saved search.</th>
</tr>
</thead>
<tbody>
<tr>
<td>You must set the title and id properties for a new saved search before you save it, either when you create it with <code>search.create(options)</code> or by setting the <code>Search.title</code> and <code>Search.id</code> properties.</td>
<td></td>
</tr>
<tr>
<td>If you do not set the saved search ID, NetSuite generates one for you. See <code>Search.id</code>.</td>
<td></td>
</tr>
<tr>
<td>Note: You do not need to set these properties if you load a previously saved search with <code>search.load(options)</code> and then save it.</td>
<td></td>
</tr>
<tr>
<td>This method also includes a promise version, <code>Search.save.promise()</code>. For more information about promises, see Promise Object.</td>
<td></td>
</tr>
</tbody>
</table>

Returns the internal ID of the saved search.
### Search.save.promise()

**Method Description**
Asynchronously saves a search created by `search.create(options)` or loaded with `search.load(options)`. Returns the internal ID of the saved search.

**Note:** For more information about using this method, see `Search.save()`. For additional information on promises, see `Promise Object`.

<table>
<thead>
<tr>
<th>Returns</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td><code>Search.save()</code></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>{1}: Missing a required argument: {2}</td>
<td>Required <code>Search.title</code> property not set on <code>search.Search</code>.</td>
</tr>
<tr>
<td>NAME_ALREADY_IN_USE</td>
<td>A search has already been saved with that name. Please use a different name.</td>
<td>The <code>Search.title</code> property on <code>search.Search</code> is not unique.</td>
</tr>
<tr>
<td>SSS_DUPLICATE_SEARCH_SCRIPT_ID</td>
<td>Saved search script IDs must be unique. Please choose another script ID. If you are trying to modify an existing saved search, use <code>search.load()</code>.</td>
<td>The <code>Search.id</code> property on <code>search.Search</code> is not unique.</td>
</tr>
</tbody>
</table>

**Syntax**

```
//Add additional code

mySalesOrderSearch.save();

//Add additional code
```

---

**Returns**

the internal search ID of the saved search as a number

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

5 units

**Module**

N/search Module

**Since**

2015.2

---

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.
<table>
<thead>
<tr>
<th><strong>Module</strong></th>
<th>N/search Module</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
search.create.promise({
  type: search.Type.SALES_ORDER
}).then(function(searchObj) {
  return searchObj.save.promise()
}).then(function (result) {
  log.debug({
    details: "Completed: " + result
  });
  // do something after completion
}).catch(function onRejected(reason) {
  // do something on rejection
});
//Add additional code
```

### Search.searchType

| **Property Description** | Internal ID name of the record type on which a search is based. Use this if you have the internal ID of the search, but do not know the record type the search was based on. For example, if the search was on a Customer record, this property is `customer`; if the search was on the Sales Order record type, this property is `salesorder`. |

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>read-only string</th>
</tr>
</thead>
</table>

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th><strong>Module</strong></th>
<th>N/search Module</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
```
```javascript
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
log.debug({
    title: 'record type: ',
    details: mySearch.searchType
});
...
//Add additional code
```

**Search.searchId**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Internal ID of the search.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The internal ID is available only when the search is either loaded with <code>search.load(options)</code> or after it has been saved with <code>Search.save()</code>. Typical values are 55 or 234 or 87, not a value like <code>customsearch_mysearch</code>. Any ID prefixed with <code>customsearch</code> is a script ID, not the internal system ID for a search.</td>
</tr>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
</tbody>
</table>

**Module**  
N/search Module

**Since**  
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
log.debug({
    title: 'search id #: ',
    details: mySearch.searchId
});
...
//Add additional code
```

**Search.filters**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Filters for the search as an array of <code>search.Filter</code> objects. Value is <code>null</code> if the search has no defined filters.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>You set this value with an array or single <code>search.Filter</code> objects to overwrite any prior filters. Use <code>null</code> to set an empty array and remove any existing filters on this search. Use <code>search.createFilter(options)</code> to create a filter.</td>
</tr>
</tbody>
</table>
**Note:** If you want to get or set a search filter expression, use the `Search.filterExpression` property.

<table>
<thead>
<tr>
<th>Type</th>
<th>search.Filter[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_SRCH_FILTER</td>
<td>An search filter contains invalid search criteria</td>
<td>Invalid value for search filter type.</td>
</tr>
</tbody>
</table>

## Syntax

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...

var myFilter = search.createFilter({
  name: 'entity',
  operator: search.Operator.ISEMPTY,
});

function createSearch() {
  var mySalesOrderSearch = search.create({
    type: search.Type.SALES_ORDER,
    filters: myFilter
  });
  ...

  //Add additional code
```

### Search.filterExpression

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Use filter expressions as a shortcut to create filters (search.Filter).</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A search filter expression is a JavaScript string array of zero or more elements. Each element is one of the following:</td>
</tr>
<tr>
<td></td>
<td>- Operator - For a list of supported operators, see <code>search.Operator</code>.</td>
</tr>
<tr>
<td></td>
<td>- Filter term</td>
</tr>
<tr>
<td></td>
<td>- Two or more filter expressions combined logically with 'and', 'or', or 'not'</td>
</tr>
<tr>
<td></td>
<td>Use null to set an empty array and remove any existing filter expressions on this search.</td>
</tr>
</tbody>
</table>
### N/search Module

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>Object[]</th>
</tr>
</thead>
</table>
| **Supported Script Types** | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| **Module** | N/search Module |
| **Since** | 2015.2 |

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
</table>
| SSS_INVALID_SRCH_FILTER_EXPR | Malformed search filter expression.  
This is a general error raised when a filter expression cannot be parsed. For example:  
```javascript
[f1, 'and', 'and', f2]
``` | The `options.filters` parameter is not a valid search filter, filter array, or filter expression. |

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
Search.create({
    type: search.Type.CUSTOMER,
    filters: [
        ['email', search.Operator.STARTSWITH, 'kwolff'], 'and', [
            ['id', search.Operator.EQUALTO, 107], 'or',
            ['id', search.Operator.EQUALTO, 2508]
        ]
    ]
});
...
//Add additional code
```

### Search.columns

| Property Description | Columns to return for this search as an array of search.Column objects or a string array of column names.  
You set this value with an array of search.Column objects or a single search.Column to overwrite any prior return columns for the search. Use null to set an empty array and remove any existing columns on this search. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>search.Column[]</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
</tbody>
</table>
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  
N/search Module

**Since**  
2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_SRCH_COLUMN</td>
<td>The value passed in was not a string or search.Column Object</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
function createSearch() {
  var mySalesOrderSearch = search.create({
    type: search.Type.SALES_ORDER,
    columns: ['entity', 'subsidiary', 'name', 'currency'],
  });
  ...
//Add additional code
```

### Search.settings

**Property Description**  
Search settings for this search as an array of search.Setting objects or a string array of column names. Search settings let you specify search parameters that are typically available only in the UI.

You set this value with an array of search.Setting objects or a single search.Setting object. You can create a search.Setting object by calling search.createSetting(options). You can also set this value with an array of column names, each of which is a string.

The supported values for a search.Setting object differ depending on the search parameter that you set. For more information, see Setting.name and Setting.value.

<table>
<thead>
<tr>
<th>Type</th>
<th>search.Setting[]</th>
<th>string[]</th>
</tr>
</thead>
</table>

**Supported Script Types**  
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  
N/search Module

**Since**  
2018.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_SRCH_SETTING</td>
<td>An unknown search parameter name is provided.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_SETTING_VALUE</td>
<td>An unsupported value is set for the provided search parameter name.</td>
</tr>
</tbody>
</table>
Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var mySearch = search.create(
    type: 'transaction',
    columns: ['trandate', 'amount', 'entity'],
    filters: [
        search.createFilter(
            name: 'internalid',
            operator: search.Operator.ANYOF,
            values: [13, 12356]
        ),
    ],
    settings: [
        search.createSetting(
            name: 'consolidationtype',
            value: 'NONE'
        )
    ]
);
...
//Add additional code
```

Search.title

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
</table>
| Search.title         | Title for a saved search. Use this property to set the title for a search before you save it for the first time. You can also set the title for a search when you create it with `search.create(options)`. The Search.title property is required to save a search with `Search.save()`.

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
</table>
| Supported Script Types | All script types For more information, see the help topic [SuiteScript 2.0 Script Types](#).
| Module   | N/search Module |
| Since    | 2015.2 |

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
function createSearch() {
    var mySalesOrderSearch = search.create(
        type: search.Type.SALES_ORDER,
        title: 'My SalesOrder Search',
    );
}
...
### Search.id

**Property Description**
Script ID for a saved search, starting with `customsearch`. If you do not set this property and then save the search, NetSuite generates a script ID for you.

**Note:** This is not the internal NetSuite ID for the saved search. See `Search.searchId`.

<table>
<thead>
<tr>
<th>Type</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/search Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
function createSearch() {
    var mySalesOrderSearch = search.create(
        type: search.Type.SALES_ORDER,
        title: 'My SalesOrder Search',
        id: 'customsearch_my_so_search',
    );
    ...
    //Add additional code
}
```

### Search.isPublic

**Property Description**
Value is `true` if the search is public, or `false` if it is not. By default, all searches created through `search.create(options)` are private.

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean <code>true</code></th>
<th><code>false</code></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
<td></td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/search Module</td>
<td></td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>
search.Result

**Object Description**
Encapsulate a single search result row. Use the methods and properties for search.Result to get the column values for the result row.

**Note:** Use search.ResultSet for the set of results from a search.

For more information about executing NetSuite searches using SuiteScript, see *Searching Overview*.

For a complete list of this object's methods and properties, see Result Object Members.

**Supported Script Types**
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/search Module

**Since**
2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
mySearch.isPublic = true;
... //Add additional code
```

```javascript
//Add additional code
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});

var searchResult = mySearch.run().getRange({
    start: 0,
    end: 100
});

for (var i = 0; i < searchResult.length; i++) {
    var entity = searchResult[i].getValue({
        name: 'entity'
    });
    var subsidiary = searchResult[i].getValue({
```
### Result.getValue(column)

**Method Description**
Used on formula and non-formula (standard) fields. Returns the string value of a specified search result column. For convenience, this method takes a single search.Column Object.

**Note:** This method is overloaded. You can also use `Result.getValue(options)` to get column values based on the name, join, and summary values for a column.

<table>
<thead>
<tr>
<th>Returns</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/search Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>search.Column</td>
<td>Required</td>
<td>The search result column from which to return a value.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var mySearch = search.load({
  id: 'customsearch_my_so_search'
});

var resultSet = mySearch.run();
var firstResult = resultSet.getRange({
  start: 0,
  end: 1
})[0];

// get the value of the second column (zero-based index)
var value = firstResult.getValue(resultSet.columns[1]);

log.debug({
  title: 'Value:',
```
Result.getValue(options)

**Method Description**
Used on formula and non-formula (standard) fields. Returns the string value of a specified search result column. Takes in arguments for name, join, and summary.

**Note:** This method is overloaded. You can also use `Result.getValue(column)` to get column values. This method takes in a single `search.Column`.

**Important:** If you have multiple search return columns and you apply grouping, all columns must include a summary property.

**Returns**
string

**Supported Script Types**
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/search Module

**Since**
2015.2

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>The search return column name.</td>
</tr>
<tr>
<td>options.join</td>
<td>string</td>
<td>Optional</td>
<td>The join id for this search return column.</td>
</tr>
<tr>
<td>options.summary</td>
<td>search.Summary</td>
<td>Optional</td>
<td>The summary type for this column. See <code>search.Summary</code>.</td>
</tr>
<tr>
<td>options.func</td>
<td>string</td>
<td>Optional</td>
<td>Special function for the search column. See <code>Column.function</code>.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

//Add additional code
...
```javascript
var searchResults = mySearch.run().getRange({
    start: 0,
    end: 100
});
for (var i = 0; i < searchResults.length; i++) {
    var amount = searchResults[i].getValue({
        name: 'amount'
    });
    var entity = searchResults[i].getValue({
        name: 'name',
        join: 'location'
    });
    ...
    //Add additional code
}
```

---

**Result.getText(column)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Used on select, image, and document fields. Returns the text value of a specified search result column. For convenience, this method takes a single search.Column Object.</th>
</tr>
</thead>
</table>

**Note:** This method is overloaded. You can also use `Result.getText(options)` to get column text value based on the name, join and summary values for a column.

<table>
<thead>
<tr>
<th>Returns</th>
<th>string</th>
</tr>
</thead>
</table>

**Supported Script Types**
All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

<table>
<thead>
<tr>
<th>Governance</th>
<th>None</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/search Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2015.2</th>
</tr>
</thead>
</table>

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>search.Column</td>
<td>Required</td>
<td>Name of the search result column.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});

var resultSet = mySearch.run();
```
```javascript
var firstResult = resultSet.getRange({
    start: 0,
    end: 1
})[0];

// get the text value of the second column (zero-based index)
var value = firstResult.getText(resultSet.columns[1]);

log.debug({
    title: 'Value: ',
    details: value
});
...
// Add additional code
```

### Result.getText(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Used on select, image, and document fields. Returns the text value of a specified search result column.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>This method is overloaded. You can also use Result.getText(column) to get a column value. This method takes in a single search.Column.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types  For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>The name of the search column.</td>
</tr>
<tr>
<td>options.join</td>
<td>string</td>
<td>Optional</td>
<td>The join internal ID for the search column.</td>
</tr>
<tr>
<td>options.summary</td>
<td>search.Summary</td>
<td>Optional</td>
<td>The summary type used for the search column. See search.Summary.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
// Add additional code
```
var searchResults = mySearch.run().getRange({
    start: 0,
    end: 100
});
for (var i = 0; i < searchResults.length; i++) {
    var amount = searchResults[i].getText({
        name: 'amount'
    });
    var entity = searchResults[i].getText({
        name: 'name',
        join: 'location'
    });
    ...
} /*Add additional code*/

---

Result.recordType

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The type of record returned in a search result row.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>search.Type enum</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
var searchResult = mySearch.run();
log.debug({
    title: 'Record Type: ',
    details: searchResult.recordType
});
... /*Add additional code*/
```

---

Result.id

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The internal ID for the record returned in a search result row.</th>
</tr>
</thead>
</table>
**Type**
number (read-only)

**Supported Script Types**
All script types
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/search Module

**Since**
2015.2

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
var mySearch = search.load({
  id: 'customsearch_my_so_search'
});

var resultSet = mySearch.run();
resultSet.each(function(result) {
  log.debug({
    title: 'Record Internal ID: ',
    details: result.id
  });
  return true;
});
...
//Add additional code
```

### Result.columns

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Array of <a href="#">search.Column</a> objects that encapsulate the columns returned in the search result row.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>search.Column[]</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/search Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
```
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});

var firstResult = mySearch.run().getRange({'
    start: 0,
    end: 1
})[0];
log.debug({'
    details: "There are " + firstResult.columns.length + " columns in the result."
});

firstResult.columns.forEach(function(col){ // log each column
    log.debug({'
        details: col
    });
});

//Add additional code

search.Column

Object Description
Encapsulates a single search column in a search.Search. Use the methods and properties available to the Column object to get or set Column properties.

You create a search column object with search.createColumn(options) and add it to a search.Search object that you create with search.create(options) or load with search.load(options).

You can pass a Column object as a parameter to the Result.getValue(column) or Result.getText(column) methods.

In addition, search.ResultSet contains an array of Column objects returned in the results of a search.

For a complete list of this object's methods and properties, see Column Object Members.

Supported Script Types
All script types

Module
N/search Module

Since
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

//Add additional code
...
search.create({
    type: search.Type.TRANSACTION,
    columns: [
        'trandate',
        'amount',
    ]
});
Column.setWhenOrderedBy(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns the search column for which the minimal or maximal value should be found when returning the <code>search.Column</code> value. For example, can be set to find the most recent or earliest date, or the largest or smallest amount for a record, and then the <code>search.Column</code> value for that record is returned.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td><code>search.Column</code></td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/search Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>The name of the search column for which the minimal or maximal value should be found.</td>
</tr>
<tr>
<td>options.join</td>
<td>string</td>
<td>Required</td>
<td>The join id for the search column.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```
// Add additional code
```
// Execute a customer search that returns the amount of the most recent sales order per customer

var filters = [];
var columns = [];
filters[0] = search.createFilter({
    name: 'recordtype',
    join: 'transaction',
    operator: search.Operator.IS,
    values: 'salesorder'
});
filters[1] = search.createFilter({
    name: 'mainline',
    join: 'transaction',
    operator: search.Operator.IS,
    values: true
});
columns[0] = search.createColumn({
    name: 'entityid',
    summary: search.Summary.GROUP
});
columns[1] = search.createColumn({
    name: 'totalamount',
    join: 'transaction',
    summary: search.Summary.MAX
});
columns[1].setWhenOrderedBy({
    name: 'trandate',
    join: 'transaction'
});
var mySearch = search.create({
    type: 'customer',
    filters: filters,
    columns: columns
});
var resultsArray = mySearch.run().getRange({
    start: 0,
    end: 100
});

// Add additional code

---

**Column.name**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Name of a search column as a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
## Column.join

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Join ID for a search column as a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>string (read-only)</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

### Syntax

```
// Add additional code
...
// Create a search definition that includes search columns
var mySearch = search.create({
    type: search.Type.CUSTOMER,
    columns: [
        search.createColumn({
            name: 'entityid'
        }),
        search.createColumn({
            name: 'email'
        })
    ]
});

// Retrieve the first search column and log its name
var myColumn = mySearch.columns[0];
log.debug(myColumn.name);

// Run the search
var results = mySearch.run();
...
// Add additional code
```

```
// Add additional code
...
log.debug({
    details: 'Join ID for Search Column: ' + columnObj.join
});
```
Column.summary

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Returns the summary type for a search column.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td><code>search.Summary</code> enum</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Module N/search Module

Since 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
... log.debug({
  details: 'Summary Type for Search Column: ' + columnObj.summary
});
... //Add additional code
```

Column.formula

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Formula used for a search column as a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To set this value, you must use formulatext, formulanumeric, formuladatetme, formulapersent, or formulacurrency.</td>
</tr>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Module N/search Module

Since 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

For example, in the UI, a field with a custom UI label named Customer Name is set by a formula of type Formula (Text) and the formula is defined with the following formula:
//Add additional code
...
var columnObj = search.createColumn({
    name: 'formulatext',
    formula: "({firstname} || ', ' || {lastname})"
});
...
//Add additional code

In the above formula, firstname and lastname are script IDs for the fields on the Customer record form.

### Column.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Label used for the search column. You can only get or set custom labels with this property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Module               | N/search Module                                                                             |
| Since                | 2015.2                                                                                       |

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var columnObj = search.createColumn({
    name: 'formulanumeric',
    label: 'Numeric Formula'
});
...
//Add additional code
```

### Column.function

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Special function applied to values in a search column. See Supported Functions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Module               | N/search Module                                                               |
Supported Functions

The following table lists the supported functions and their internal IDs:

<table>
<thead>
<tr>
<th>Internal ID</th>
<th>Name</th>
<th>Date Function</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>percentOfTotal</td>
<td>% of Total</td>
<td>No</td>
<td>percent</td>
</tr>
<tr>
<td>absoluteValue</td>
<td>Absolute Value</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>ageInDays</td>
<td>Age In Days</td>
<td>Yes</td>
<td>integer</td>
</tr>
<tr>
<td>ageInHours</td>
<td>Age In Hours</td>
<td>Yes</td>
<td>integer</td>
</tr>
<tr>
<td>ageInMonths</td>
<td>Age In Months</td>
<td>Yes</td>
<td>integer</td>
</tr>
<tr>
<td>ageInWeeks</td>
<td>Age In Weeks</td>
<td>Yes</td>
<td>integer</td>
</tr>
<tr>
<td>ageInYears</td>
<td>Age In Years</td>
<td>Yes</td>
<td>integer</td>
</tr>
<tr>
<td>calendarWeek</td>
<td>Calendar Week</td>
<td>Yes</td>
<td>date</td>
</tr>
<tr>
<td>day</td>
<td>Day</td>
<td>Yes</td>
<td>date</td>
</tr>
<tr>
<td>month</td>
<td>Month</td>
<td>Yes</td>
<td>text</td>
</tr>
<tr>
<td>negate</td>
<td>Negate</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>numberAsTime</td>
<td>Number as Time</td>
<td>No</td>
<td>text</td>
</tr>
<tr>
<td>quarter</td>
<td>Quarter</td>
<td>Yes</td>
<td>text</td>
</tr>
<tr>
<td>rank</td>
<td>Rank</td>
<td>No</td>
<td>integer</td>
</tr>
<tr>
<td>round</td>
<td>Round</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>roundToHundredths</td>
<td>Round to Hundredths</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>roundToTenths</td>
<td>Round to Tenths</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>weekOfYear</td>
<td>Week of Year</td>
<td>Yes</td>
<td>text</td>
</tr>
<tr>
<td>year</td>
<td>Year</td>
<td>Yes</td>
<td>text</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_SRCH_FUNCTN</td>
<td>A search.Column contains an invalid function: {1}.</td>
<td>Unknown function is set.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
```

SuiteScript 2.0 API Reference
```javascript
var columnObj = search.createColumn({
    name: 'trandate',
    function: 'ageInDays'
});
```

### Column.sort

**Property Description**

The sort order of the column.

Use the `search.Sort` enum to set the value.

If `Column.sort` is not set, the column is not sorted in any particular order.

**Type**

`search.Sort` enum

**Supported Script Types**

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/search Module

**Since**

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code

var columnObj = search.createColumn({
    name: 'invoice',
    sort: search.Sort.DESC
});
```

### search.Filter

**Object Description**

Encapsulates a search filter used in a search. Use the properties for the Filter object to get and set the filter properties.

You create a search filter object with `search.createFilter(options)` and add it to a `search.Search` object that you create with `search.create(options)` or load with `search.load(options)`.

**Note:** NetSuite uses an implicit `AND` operator with search filters, as opposed to filter expressions which explicitly use either `AND` and `OR` operators.

Use the following guidelines with the Filter object:
To search for a “none of null” value, meaning do not show results without a value for the specified field, use a value of @NONE@ in the `Filter.formula` property.

To search on checkbox fields, use the `IS` operator with a value of T or F to search for checked or unchecked fields, respectively.

For a complete list of this object’s methods and properties, see `Filter Object Members`.

**Supported Script Types**

`All script types`

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Module**

`N/search Module`

**Since**

`2015.2`

---

**Filter.name**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Name or internal ID of the search field as a string. For more information, see <code>search.createFilter(options)</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types For more information, see the help topic <code>SuiteScript 2.0 Script Types</code>.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td><code>N/search Module</code></td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td><code>2015.2</code></td>
</tr>
</tbody>
</table>

---

**Syntax**

```javascript
//Add additional code
...
var mySearchFilter = search.createFilter({
  name: 'entity',
  operator: search.Operator.ISEMPTY,
});
...
//Add additional code
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/search Module Script Samples`.

---

```javascript
//Add additional code
...
log.debug({
  details: 'Filter Name: ' + filterObj.name
});
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/search Module Script Samples`.
Filter.join

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Join ID for the search filter as a string.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Module | N/search Module |
Since  | 2015.2 |

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
// Add additional code
...
// Create a filter joined to another record type. When you create a joined filter:
// - The name property is the field ID of the field in the joined record that you are filtering on
// - The join property is the field ID of the field in the current record that contains the record type you want to join to
// - The operator property is the operator to use to filter the results
// - The values property contains the values to use to filter the results
search.createFilter({
    name: 'joined_record_field_id',
    join: 'current_record_field_id',
    operator: search.Operator.IS,
    values: ['valueToFilter']
});
// Add additional code
```

Filter.operator

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Operator used for the search filter. This value is set with the search.Operator enum. The search.Operator enum contains the valid operator values for this property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types                                                                  For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Module | N/search Module |
Since  | 2015.2 |
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
// Add additional code
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISEMPTY
});
log.debug({
    details: 'Operator Used: ' + mySearchFilter.operator
});
...
// Add additional code
```

**Filter.summary**

**Property Description**
Summary type for the search filter. Use this property to get or set the value of the summary type. See `search.Summary`.

**Type**
`search.Summary`

**Supported Script Types**
All script types
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/search Module

**Since**
2015.2

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_INVALID_SRCH_FILTER_SUM</td>
<td>A search.Filter contains an invalid summary type: <code>{1}</code>.</td>
<td>Unknown summary type is set.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
// Add additional code
...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISNOTEMPTY,
});
```
Filter.formula

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Formula used by the search filter. Use this property to get or set the formula used by the search filter. For more information about the formula property, see search.createFilter(options).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
log.debug(
  {
    details: 'Search Filter Formula: ' + filterObj.formula
  });
...
//Add additional code
```

search.ResultSet

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates a set of search results returned by Search.run(). Use the methods and properties for the ResultSet object to iterate through each result returned by the search or access an arbitrary slice of results, up to 1000 results at a time. For a complete list of this object's methods and properties, see ResultSet Object Members.</th>
</tr>
</thead>
</table>

⚠️ Important: Search result sets are not cached. If records applicable to your search are created, modified, or deleted at the same time you are traversing your result set, your result set may change.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>All script types</th>
</tr>
</thead>
</table>
For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Module</th>
<th>N/search Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
// Add additional code
...
// Load a saved search. Alternatively, you can create a search using search.create(options)
// and other methods in the N/search module.
var mySearch = search.load({
  id: 'customsearch_my_cs_search'
});

// Run the search, and use ResultSet.each(callback) to define a callback function to
// execute on each search result.
//
// In this example, the saved search that was loaded above searches for Customer records.
// The Result.getValue(options) method obtains the search result value of one of the search
// columns that was specified in the search definition. Both 'entityid' and 'email' are valid
// search column names for a Customer record.
mySearch.run().each(function(result) {
  var entity = result.getValue({
    name: 'entityid'
  });
  log.debug(entity);
  var email = result.getValue({
    name: 'email'
  });
  log.debug(email);
  return true;
});
...
// Add additional code
```

**ResultSet.getRange(options)**

**Method Description** Retrieve a slice of the search result as an array of search.Result objects.

The start parameter is the inclusive index of the first result to return. The end parameter is the exclusive index of the last result to return. For example, getRange(0, 10) retrieves 10 search results, at index 0 through index 9. Unlimited rows in the result are supported, however you can only return 1,000 at a time based on the index values.

If there are fewer results available than requested, then the array will contain fewer than end - start entries. For example, if there are only 25 search results, then getRange(20, 30) will return an array of 5 search.Result objects.
The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.start</td>
<td>number</td>
<td>Required</td>
<td>Index number of the first result to return, inclusive.</td>
</tr>
<tr>
<td>options.end</td>
<td>number</td>
<td>Required</td>
<td>Index number of the last result to return, exclusive.</td>
</tr>
</tbody>
</table>

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var results = rs.getRange({
    start: 0,
    end: 1000
});
 ...
//Add additional code
```

**ResultSet.getRange.promise(options)**

Method used to asynchronously retrieve a slice of the search result as an array of `search.Result` objects.

**Note:** For information about the parameters and errors thrown for this method, see `ResultSet.getRange(options)`. For additional information on promises, see Promise Object.  

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.Result[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>10 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
</tbody>
</table>
## ResultSet.each(callback)

**Method Description**  Use a developer-defined function to invoke on each row in the search results, up to 4000 results at a time. The callback function must use the following signature:

```javascript
boolean callback(search.Result result);
```

The callback function takes a `search.Result` object as an input parameter and returns a boolean which can be used to stop the iteration with a value of `false`, or continue the iteration with a value of `true`.

**Important:** The work done in the context of the callback function counts towards the governance of the script that called it. For example, if the callback function is running in the context of a scheduled script, which has a 10,000 unit governance limit, make sure the amount of processing within the callback function does not put the entire script at risk of exceeding scheduled script governance limits.

**Returns**  void

**Supported Script Types**  All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**  10 units
**Parameters**

⚠️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td>function</td>
<td>Required</td>
<td>Named JavaScript function or anonymous inline function that contains the logic to process a <code>search.Result</code> object.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
mySearch.run().each(function(result) {
    var entity = result.getValue({'
        name: 'entity'
    });
    var subsidiary = result.getValue({'
        name: 'subsidiary'
    });
    return true;
});
...
//Add additional code
```

**ResultSet.each.promise(callback)**

**Method Description**

Asynchronously uses a developer-defined function to invoke on each row in the search results, up to 4000 results at a time. The callback function must use the following signature:

⚠️ **Note:** For information about the parameters and errors thrown for this method, see `ResultSet.each(callback)`. For additional information on promises, see [Promise Object](#).

<table>
<thead>
<tr>
<th>Returns</th>
<th>void</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>10 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```javascript
//Add additional code
...
mySearch.run().each.promise(function(result) {
    var entity = result.getValue({
        name: 'entity'
    });
    var subsidiary = result.getValue({
        name: 'subsidiary'
    });
    return true;
})
.then(function(response){
    log.debug({
        title: 'Completed',
        details: response
    });
})
.catch(function onRejected(reason) {
    log.debug({
        title: 'Failed: ',
        details: reason
    });
})
...
//Add additional code
```

### ResultSet.columns

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>An array of <code>search.Column</code> objects that represent the columns returned in the search results.</td>
<td></td>
</tr>
</tbody>
</table>

**Type**

`search.Column[]`

This property is read-only

**Supported Script Types**

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

`N/search Module`

**Since**

2015.2

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
```
... 
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});

var resultSet = mySearch.run();
log.debug({
    details: "There are " + resultSet.columns.length + " columns in the result set:'
});

resultSet.columns.forEach(function(col){ // log each column
    log.debug({
        details: col
    });
});
...
//Add additional code

search.Page

Object Description
Encapsulates an individual search page containing a result set for a paginated search.
For a complete list of this object's methods and properties, see Page Object Members.

Supported Script Types
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/search Module

Since
Version 2015 Release 1

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```
//Add additional code
...
var page = pagedData.fetch({
    index: lastPageRange.index
});
...
//Add additional code
```

Page.next()

Method Description
Method used to fetch the next segment of data (bounded by search.PageRange).
Moves the current page to next range.
### Returns
Void

### Supported Script Types
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

### Governance
5 units

### Module
N/search Module

### Since
2016.1

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_PAGE_RANGE</td>
<td>Invalid page range.</td>
<td>The page range is invalid, or when the page is the last page.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
while (!page.isFirst){
  page = page.next();
  ...
  //Add additional code
```

### Page.next.promise()

**Method Description**
Method used to asynchronously fetch the next segment of data (bounded by search.PageRange).

Moves the current page to another range. The promise is complete when the data for this range is loaded or rejected.

**Note:** For information about errors thrown for this method, see Page.next(). For additional information on promises, see Promise Object.

<table>
<thead>
<tr>
<th>Returns</th>
<th>Void</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td>Page.next()</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
<td></td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see **Promise Object**.

```javascript
// Add additional code
...
// In this snippet, myPage is a Page object that encapsulates a page of search results,
// and processPage is the name of a callback function to execute when the promise
// method returns
return myPage.next.promise().then(processPage);
...
// Add additional code
```

## Page.prev()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Moves the current page to previous range.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>Void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a></td>
</tr>
<tr>
<td>Module</td>
<td><a href="#">N/search Module</a></td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_PAGE_RANGE</td>
<td>Invalid page range.</td>
<td>The page range is invalid, or when the page is the first page.</td>
</tr>
</tbody>
</table>

## Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
// Add additional code
...
while (!page.isLast){
    page = page.prev();
    ...
// Add additional code
```

## Page.prev.promise()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Method used to asynchronously fetch the previous segment of data (bounded by search.PageRange).</th>
</tr>
</thead>
</table>
Moves the current page to another range. The promise is complete when the data for this range is loaded or rejected.

**Note:** For information about errors thrown for this method, see `Page.prev()`. For additional information on promises, see Promise Object.

- **Returns**: Void
- **Synchronous Version**: `Page.prev()`
- **Supported Script Types**: All client-side scripts
  For more information, see the help topic SuiteScript 2.0 Client Script Type.
- **Governance**: 5 units
- **Module**: N/search Module
- **Since**: 2016.1

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
return mypage.prev().promise().then(processPage);
...
//Add additional code
```

### Page.data

**Property Description**: The results from a paginated search.

**Type**: `search.Result[]`

This property is read-only.

**Supported Script Types**: All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**: N/search Module

**Since**: 2016.1

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
```
function processPage(page){
    page.data.forEach(function(value){
        log.debug({
            details: "data: " + page.data
        });
    });
    ...
    //Add additional code
}

Page.isFirst

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether the page is within the first range of the result set. Flags the start of the data collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean true</td>
</tr>
<tr>
<td></td>
<td>This property is read-only.</td>
</tr>
</tbody>
</table>

Supported Script Types

- All script types
- For more information, see the help topic SuiteScript 2.0 Script Types.

Module

- N/search Module

Since

- 2016.1

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```
//Add additional code
...
while (!page.isFirst){
    page = page.next();
    ...
    //Add additional code
```

Page.isLast

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether a page is within the last range of the result set. Flags the end of the data collection.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean true</td>
</tr>
<tr>
<td></td>
<td>This property is read-only.</td>
</tr>
</tbody>
</table>

Supported Script Types

- All script types
- For more information, see the help topic SuiteScript 2.0 Script Types.

Module

- N/search Module

Since

- 2016.1
Syntax

```javascript
//Add additional code
...
while (!page.isLast){
    page = page.prev();
    ...
//Add additional code
```

### Page.pagedData

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page.pagedData</td>
<td>The <code>PagedData</code> object used to fetch this <code>Page</code> object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>search.PagedData</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
</tbody>
</table>

**Module**  
N/search Module

**Since**  
2016.1

### Page.pageRange

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page.pageRange</td>
<td>The <code>PageRange</code> object used to fetch this <code>Page</code> object. Page boundary information with the key and label.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>search.PageRange</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
</tbody>
</table>

**Module**  
N/search Module
search.PagedData

Object Description

Holds metadata for a paginated query.

This object provides a high-level view of a search result, giving the total count of records, a list of pages ranges, and page size.

For a complete list of this object's methods and properties, see PagedData Object Members.

Important: Search result sets are not cached. If records applicable to your search are created, modified, or deleted at the same time you are traversing your result set, your result set may change.

Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/search Module

Since

Version 2015 Release 1

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
// Add additional code
...
log.debug({
  details: "Page Range: " + mySearchPage.pageRange
});
...
// Add additional code
```

```javascript
// Add additional code
... // Run the paged search
var pagedData = mySearch.runPaged({
  pageSize: 1000
});
...
// Use the count property to count the
// search results easily
```
var resultCount = mySearch.runPaged({
  pageSize: 1000
}).count;
...
//Add additional code

PagedData.fetch(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>This method retrieves the data within the specified page range. This method also includes a promise version, PagedData.fetch.promise(). For more information about promises, see Promise Object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>search.Page</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pageRange.index</td>
<td>number</td>
<td>required</td>
<td>The index of the page range that bounds the desired data.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_PAGE_RANGE</td>
<td>Invalid page range.</td>
<td>The page range is not valid.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var page = pagedData.fetch({
  index: lastPageRange.index
});
...
//Add additional code
```
PagedData.fetch.promise()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>This method asynchronously retrieves the data bounded by the pageRange parameter.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>For information about the parameters and errors thrown for this method, see PagedData.fetch(options). For additional information on promises, see Promise Object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td>PagedData.fetch(options)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
return pagedData.fetch.promise().then(processPage);
...
//Add additional code
```

PagedData.count

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The total number of results when Search.runPaged(options) was executed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td></td>
<td>This property is read-only.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.
PagedData.pageRanges

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The collection of PageRange objects that divide the entire result set into smaller groups. Includes page range information with the key and label for rendering.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td><code>search.PageRange[]</code></td>
</tr>
<tr>
<td></td>
<td>This property is read-only.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic [SuiteScript 2.0 Script Types](<a href="https://oracle">https://oracle</a> netsuite.com).</td>
</tr>
</tbody>
</table>

Module | N/search Module
Since  | 2016.1

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](https://oracle netsuite.com).

```javascript
//Add additional code ...
log.debug({
    details: "Result Count: " + myPagedData.count
});
...
//Add additional code
```

PagedData.pageSize

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Maximum number of entries per page Possible values are 5 - 1000 entries per page.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td><code>number</code></td>
</tr>
<tr>
<td></td>
<td>This is a read-only property.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic [SuiteScript 2.0 Script Types](<a href="https://oracle">https://oracle</a> netsuite.com).</td>
</tr>
</tbody>
</table>
**Module**
N/search Module

**Since**
2016.1

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
log.debug({
    details: 'Max Page Size: ' + myPagedData.pageSize
});
...
//Add additional code
```

### PagedData.searchDefinition

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The search criteria used to execute the result set for this PagedData Object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>read-only search/Search</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

**Module**
N/search Module

**Since**
2016.1

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
log.debug({
    details: "Search Details: " + myPagedData.searchDefinition
});
...
//Add additional code
```

### search.PageRange

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Defines the page range to contain the result set For a complete list of this object's properties, see PageRange Object Members.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>
## PageRange.compoundLabel

**Property Description**  
Human-readable label with beginning and ending range identifiers

**Type**  
read-only string

**Supported Script Types**  
All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](https://oracle-netSuite.github.io/)

### Syntax

```
//Add additional code
...
var page = pagedData.fetch({
  index: lastPageRange
});
...
//Add additional code
```

### PageRange.index

**Property Description**  
The index of the pageRange

**Type**  
number

This property is read-only.

**Supported Script Types**  
All script types
## search.Setting

**Object Description**  Defines a search setting.

Search settings let you specify search parameters that are typically available only in the UI. The following settings are supported:

- **Consolidated Exchange Rate**: This setting affects how consolidation is performed (for example, consolidation using the Average rate type, consolidation using the Historical rate type, and so on). This setting applies to transaction searches, and it is applicable only to OneWorld accounts.
- **Show Period End Transactions**: This setting indicates whether period end transactions are included in search results. This setting applies to transaction searches, and it is applicable only to OneWorld accounts. It also requires the Show Period End transactions feature to be enabled.

Use `search.createSetting(options)` to create a setting. After you create your settings, assign them as array values to `Search.settings`. For a complete list of this object's properties, see Setting Object Members.
var mySearch = search.create(
  type: 'transaction',
  columns: ['trandate', 'amount', 'entity'],
  filters: [
    search.createFilter({
      name: 'internalid',
      operator: search.Operator.ANYOF,
      values: [13, 12356]
    }),
    search.createSetting({
      name: 'consolidationtype',
      value: 'NONE'
    })
  ]
);
Setting.value

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The value of the search parameter.</td>
<td></td>
</tr>
</tbody>
</table>

This property is set when you call `search.createSetting(options)`. If you specify `consolidationtype` as the search parameter name (`Setting.name`), the following values are supported for this parameter:

- ACCTTYPE
- AVERAGE
- CURRENT
- HISTORICAL
- NONE

If you specify `includeperiodendtransactions` as the search parameter name (`Setting.name`), the following values are supported for this parameter:

- F
- FALSE
- T
- TRUE

These values are not case sensitive.

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>This property is read-only.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>All script types</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/search Module</th>
</tr>
</thead>
</table>

| Since | 2018.2 |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...```
var mySearch = search.create(
  type: 'transaction',
  columns: [ 'trandate', 'amount', 'entity' ],
  filters: [
    search.createFilter({
      name: 'internalid',
      operator: search.Operator.ANYOF,
      values: [13, 12356]
    })],
  settings: [
    search.createSetting({
      name: 'consolidationtype',
      value: 'NONE'
    })]
});
...  
//Add additional code

search.create(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a new search and returns it as a <code>search.Search</code> object.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The search can be modified and run as an on demand search with <code>Search.run()</code>, without saving it. Alternatively, calling <code>search.save()</code> will save the search to the database, so it can be reused later in the UI or loaded with <code>search.load(options)</code>.</td>
</tr>
<tr>
<td>Note:</td>
<td>This method is agnostic in terms of its <code>options.filters</code> argument. It can accept input of a single <code>search.Filter</code> object, an array of <code>search.Filter</code> objects, or a search filter expression.</td>
</tr>
</tbody>
</table>

The `search.create(options)` method also includes a promise version, `search.create.promise(options)`. For more information about promises, see Promise Object.

<table>
<thead>
<tr>
<th>Important:</th>
<th>When you use this method to create a search, consider the following:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>- When you define the search, make sure you sort using the field with the most unique values, or sort using multiple fields. Sorting with a single field that has multiple identical values can cause the result rows to be in a different order each time the search is run.</td>
</tr>
<tr>
<td></td>
<td>- You cannot directly create a filter or column for a list/record type field in SuiteScript by passing in its text value. You must use the field's internal ID. If you must use the field's text value, you can create a filter or column with a formula using <code>name: 'formulatext'</code>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th><code>search.Search</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

For more information, see the help topic SuiteScript 2.0 Script Types.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>string</td>
<td>Required</td>
<td>The search type that you want to base the search on. Use the search.Type enum for this argument.</td>
</tr>
<tr>
<td>options.filters</td>
<td>search.Filter[] / Object[]</td>
<td>Optional</td>
<td>A single search.Filter object, an array of search.Filter objects, a search filter expression, or an array of search filter expressions. A search filter expression can be passed in as an Object with the following properties:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>name (required)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>join</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>operator (required)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>summary</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>formula</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information about these properties, see Filter Object Members. If a provided filter value has an incorrect type (for example, a string instead of a number), the filter value is ignored. For server-side scripts, a log entry is created when an incorrect type is provided.</td>
</tr>
<tr>
<td>options.filterExpression</td>
<td>Object[]</td>
<td>Optional</td>
<td>Search filter expression for the search as an array of expression objects. A search filter expression is a JavaScript string array of zero or more elements. Each element is one of the following:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Operator - either 'NOT', 'AND', or 'OR'</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Filter term</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Nested search filter expression</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You set this value with an array of expression objects or single filter expression object to overwrite any prior filter expressions. Use null to set an empty array and remove any existing filter expressions on this search.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Note:</strong> If you want to get or set a search filters, use the Search.filters property.</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.settings| search.Setting | Optional            | Search settings for this search as a single search.Setting object or an array of search.Setting objects. Search settings let you specify search parameters that are typically available only in the UI. See Search.settings. You can optionally pass in an Object or array of Objects with the following properties to represent a setting:  
  - name  
  - value  
  For more information about these properties, see Setting Object Members.                                                                                                                             |
| options.title   | string      | Optional            | The name for a saved search. The title property is required to save a search with Search.save().                                                                                                                                                                                                                                                |
| options.id      | string      | Optional            | Script ID for a saved search. If you do not set the saved search ID, NetSuite generates one for you. See Search.id.                                                                                                                                                                                                                             |
| options.isPublic| boolean     | Optional            | Set to true to make the search public. Otherwise, set to false. If you do not set this parameter, it defaults to false. This parameter sets the value for the Search.isPublic property.                                                                                                                                               |

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>Required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_FILTER_EXPR</td>
<td>The options.filters parameter is not a valid search filter, filter array, or filter expression.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_COL</td>
<td>The options.columns parameter is not a valid column, string, or column or string array.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_SETTING</td>
<td>An unknown search parameter name is provided.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_SETTING_VALUE</td>
<td>An unsupported value is set for the provided search parameter name.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var mySalesOrderSearch = search.create({
    type: search.Type.SALES_ORDER,
    title: 'My Second SalesOrder Search',
    id: 'customsearch_my_second_so_search',
    columns: [{
        name: 'entity'
    }, {
        name: 'subsidiary'
    }, {
        name: 'name'
    }, {
        name: 'currency'
    }],
    filters: [{
        name: 'mainline',
        operator: 'is',
        values: ['T']
    }],
    settings: [{
        name: 'consolidationtype',
        value: 'AVERAGE'
    }]
});
...
//Add additional code
```

**search.create.promise(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a new search asynchronously and returns it as a <code>search.Search</code> object.</th>
</tr>
</thead>
</table>

**Note:** For information about the parameters and errors thrown for this method, see `search.create(options)`. For additional information on promises, see Promise Object.

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td><code>search.create(options)</code></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
**search.createSetting(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a new search setting and returns it as a search.Setting object.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Search settings let you specify search parameters that are typically available only in the UI. The following settings are supported:</td>
</tr>
<tr>
<td></td>
<td>- <strong>Consolidated Exchange Rate</strong>: This setting affects how consolidation is performed (for example, consolidation using the Average rate type, consolidation using the Historical rate type, and so on). This setting applies to transaction searches, and it is applicable only to OneWorld accounts.</td>
</tr>
<tr>
<td></td>
<td>- <strong>Show Period End Transactions</strong>: This setting indicates whether period end transactions are included in search results. This setting applies to transaction searches, and it is applicable only to OneWorld accounts. It also requires the Period End Journal Entries feature to be enabled.</td>
</tr>
<tr>
<td></td>
<td>After you create your settings, assign them as array values to Search.settings.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.Setting</th>
</tr>
</thead>
</table>

| Supported Script Types | All script types |

| Governance | None |

| Module | N/search Module |

| Since | 2018.2 |
## Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
</table>
| options.name      | string    | Required            | The name of the search parameter to set. This value sets the `Setting.name` property. Use one of the following values for this parameter:  
  - consolidationtype: This value corresponds to the Consolidated Exchange Rate setting.  
  - includeperiodendtransactions: This value corresponds to the Show Period End Transactions setting.                                                                                                                                                                                                                                                                                                             | 2018.2 |
| options.value     | string    | Required            | The value of the search parameter. If you are executing a joined search, this value is the join ID used for the search field specified by the `options.name` parameter. This value sets the `Setting.value` property. If you specify `consolidationtype` as the search parameter name, use one of the following values for this parameter:  
  - ACCTTYPE  
  - AVERAGE  
  - CURRENT  
  - HISTORICAL  
  - NONE  
  The default value is ACCTTYPE, which represents the type of consolidation associated with the account. If you specify `includeperiodendtransactions` as the search parameter name, use one of the following values for this parameter:  
  - F  
  - FALSE  
  - T  
  - TRUE  
  The default value is false. These values are not case sensitive.                                                                                                                                                                                                                                                                                                           | 2018.2 |

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_SETTING</td>
<td>An unknown search parameter name is provided.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_SETTING_VALUE</td>
<td>An unsupported value is set for the provided search parameter name.</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
var mySearch = search.create({
  type: 'transaction',
  columns: [ 'trandate', 'amount', 'entity' ],
  filters: [
    search.createFilter({
      name: 'internalid',
      operator: search.Operator.ANYOF,
      values: [13, 12356]
    })),
  settings: [ search.createSetting({
      name: 'consolidationtype',
      value: 'NONE'
    })]
});
...
//Add additional code
```

#### search.load(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Loads an existing saved search and returns it as a <code>search.Search</code>. The saved search could have been created using the UI or created with <code>search.create(options)</code> and <code>Search.save()</code>. The <code>search.load(options)</code> method also includes a promise version, <code>search.load.promise(options)</code>. For more information about promises, see <a href="#">Promise Object</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td><code>search.Search</code></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>Required</td>
<td>Internal ID or script ID of a saved search. The script ID starts with <code>customsearch</code>. See <code>Search.id</code>.</td>
<td>2015.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>--------------</td>
<td>---------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>Required if the saved search to load uses a standalone search type, optional otherwise</td>
<td>The search type of the saved search to load. Use a value from the search.Type enum for this parameter. This parameter is required if the saved search to load uses a standalone search type. A standalone search type is a search type that does not have a corresponding record type. Typically, the search type of the saved search can be determined automatically based on the corresponding record type. In this case, this parameter is not required. For standalone search types, you must specify the search type explicitly using this parameter. The following is a list of standalone search types: DeletedRecord, EndToEndTime, ExpenseAmortPlanAndSchedule, RevRecPlanAndSchedule, GLinesAuditLog, Crosschargeable, FinRptAggregateFR, BillingAccountBillCycle, BillingAccountBillRequest, BinItemBalance, PaymentEvent, Permission, GatewayNotification, TimeApproval, RecentRecord, Role, SavedSearch, ShoppingCart, SubscriptionRenewalHistory, SuiteScriptDetail, SupplyChainSnapshotDetails, SystemNote, TaxDetail, TimesheetApproval, Uber, ResAllocationTimeOffConflict, ComSearchOneWaySyn, ComSearchGroupSyn, Installment, InventoryBalance, InventoryNumberBin, InventoryNumberItem</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Parameter, Type, Required / Optional, Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>■ InventoryStatusLocation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ InvtNumberItemBalance</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>■ ItemBinNumber</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>(1): Missing a required argument: (2)</td>
<td>Required parameter is missing.</td>
</tr>
<tr>
<td>INVALID_SEARCH</td>
<td>That search or mass update does not exist.</td>
<td>Cannot find saved search with the saved search ID from options.id parameter.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var mySearch = search.load({
    id: 'customsearch_my_so_search'
});
...
//Add additional code
```

**search.load.promise(options)**

**Method Description**

Loads an existing saved search asynchronously and returns it as a `search.Search` object. The saved search could have been created using the UI or created with `search.create(options)` and `Search.save()`.

**Note:** For information about the parameters and errors thrown for this method, see `search.load(options)`. For additional information on promises, see Promise Object.

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td>search.load(options)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>5 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see [Promise Object](#).

```javascript
// Add additional code
...
search.load.promise({
  type: search.Type.SALES_ORDER,
  id: 'customsearch_txn_search_salesorder'
}).then(function (result) {
  log.debug({
    details: "Completed: ' + result
  });
  // do something after completion
}).catch(function onRejected(reason) {
  // do something on rejection
});
...
// Add additional code
```

---

**search.delete(options)**

| Method Description | Deletes an existing saved search. The saved search could have been created using the UI or created with `search.create(options)` and `Search.save()`. The `search.delete(options)` method also includes a promise version, `search.delete.promise(options)`. For more information about promises, see [Promise Object](#).
| Returns | void |
| Supported Script Types | All script types<br>For more information, see the help topic [SuiteScript 2.0 Script Types](#).
| Governance | 5 units |
| Module | N/search Module |
| Since | 2015.2 |

**Parameters**

**Note:** The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>Required</td>
<td>Internal ID or script ID of a saved search. The script ID starts with <code>customsearch</code>. See <code>Search.id</code>.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>(1): Missing a required argument: (2)</td>
<td>Required parameter is missing.</td>
</tr>
<tr>
<td>INVALID_SEARCH</td>
<td>That search or mass update does not exist.</td>
<td>Cannot find saved search with the saved search ID from options.id parameter.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
search.delete({
  id: 'customsearch_my_so_search'
});
...
//Add additional code
```

search.delete.promise(options)

Method Description

Deletes an existing saved search asynchronously and returns it as a search.Search object. The saved search can be created using the UI or created with search.create(options) and Search.save().

Note: For information about the parameters and errors thrown for this method, see search.delete(options). For additional information on promises, see Promise Object.

Returns void
Synchronous Version search.delete(options)
Supported Script Types All client-side scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.
Governance 5 units
Module N/search Module
Since 2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
```
search.duplicates(options)

**Method Description**
Perform a search for duplicate records based on the account's duplicate detection configuration.

This method also includes a promise version, `search.duplicates.promise(options)`. For more information about promises, see Promise Object.

⚠️ **Important:** This API works only for records that support duplicate record detection (for example, customer, lead, prospect, contact, partner, and vendor records).

For more information about duplicate record detection, see the help topic Duplicate Record Detection.

**Returns**
- `search.Result[]` that contains the duplicate records
- Results are limited to 1000 rows.
- If there are no search results, this method returns `null`.

**Supported Script Types**
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
10 units

**Module**
N/search Module

**Since**
2015.2

### 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>options.type</td>
<td>enum</td>
<td>Required</td>
<td>The search type that you want to check for duplicates. Use the <code>search.Type</code> enum for this parameter. The type you specify must correspond to a record type that supports duplicate record detection (for example, customer, lead, prospect, contact, partner, and vendor records).</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Parameter Reference

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>customer, lead, prospect, contact, partner, and vendor records.</td>
<td></td>
</tr>
<tr>
<td>options.fields</td>
<td>Object</td>
<td>Optional</td>
<td>A set of key/value pairs used to detect duplicates. The keys are internal ID names of the fields used to detect duplicates. You can specify fields such as companyname, email, name, phone, address1, city, state, and zipcode. For example, to detect duplicates based on the value of the email field, use 'email': '<a href="mailto:sample@test.com">sample@test.com</a>'. Use this parameter to specify the fields (and their values) to use to detect duplicates. If you are searching for duplicates based on fields that appear on a certain record type, this parameter is required. If you are searching for duplicates of a specific record (of the specified type), use the options.id parameter instead.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.id</td>
<td>number</td>
<td>Optional</td>
<td>Internal ID of an existing record. Use this parameter to specify a record to detect duplicates of. The duplicate record detection settings in the account determine which fields are used to detect duplicates of the specified record. If you are searching for duplicates of a specific record (of the specified type), this parameter is required. If you are searching for duplicates based on fields that appear on a certain record type, use the options.fields parameter instead.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>(1): Missing a required argument: (2)</td>
<td>Required parameter is missing.</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code...

// Search for duplicates of a specific record using the options.id parameter
var duplicatesOfRecord = search.duplicates({
  type: search.Type.CONTACT,
  id: 425
});

// Search for duplicates based on specific fields on a record type using the options.fields parameter
var duplicatesUsingFields = search.duplicates({
  type: search.Type.CONTACT,
  fields: {
```
search.duplicates.promise(options)

**Method Description**
Performs a search for duplicate records asynchronously based on the Duplicate Detection configuration for the account. Returns an array of search.Result objects. This method only applies to records that support duplicate record detection. These records include customer | lead | prospect | partner | vendor | contact.

**Note:** For information about the parameters and errors thrown for this method, see search.duplicates(options). For additional information on promises, see Promise Object.

**Returns**
search.Result[]

**Synchronous Version**
search.duplicates(options)

**Supported Script Types**
All client-side scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**
10 units

**Module**
N/search Module

**Since**
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
search.duplicates.promise({
    type: search.Type.CUSTOMER,
    id: 28
})
  .then(function (result) {
    log.debug({
        details: "Completed: " + result
    });
    // do something after completion
  })
  .catch(function onRejected(reason) {
    // do something on rejection
  });
...
//Add additional code
```
search.global(options)

**Method Description**

Performs a global search against a single keyword or multiple keywords.

Similar to the global search functionality in the UI, you can programmatically filter the global search results that are returned. For example, you can use the following filter to limit the returned records to Customer records:

`'cu: simpson'`

The `search.global(options)` method also includes a promise version, `search.global.promise(options)`. For more information about promises, see [Promise Object](#).

For more information about global search, see the help topic [Global Search](#).

**Returns**

`search.Result[]` as an array of result objects containing these columns: name, type, info1, and info2

Results are limited to 1000 records.

If there are no search results, this method returns `null`.

**Supported Script Types**

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

10 units

**Module**

N/search Module

**Since**

2015.2

### 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>options.keywords</td>
<td>string</td>
<td>Required</td>
<td>Global search keywords string or expression.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>(1): Missing a required argument: (2)</td>
<td>Required parameter is missing.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
var customerSearch = search.global({
    keywords: 'cu: simpson'
});
...
```
search.global.promise(options)

| Method Description | Performs a global search asynchronously against a single keyword or multiple keywords. Returns an array of search.Result objects with four columns: name, type, info1, and info2. |

**Note:** For information about the parameters and errors thrown for this method, see search.global(options). For additional information on promises, see Promise Object.

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.Result[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous Version</td>
<td>search.global(options)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All client-side scripts For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
</tr>
<tr>
<td>Governance</td>
<td>10 units</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.

```javascript
//Add additional code
...
search.global.promise({
    keywords: 'Alan Rath'
})
.then(function (result) {
    log.debug(
        {details: "Completed: " + result}
    );
    // do something after completion
}
.catch(function onRejected(reason) {
    // do something on rejection
});
...
//Add additional code
```

search.lookupFields(options)

| Method Description | Performs a search for one or more body fields on a record. |

SuiteScript 2.0 API Reference
You can use joined-field lookups with this method, with the following syntax:

```
join_id.field_name
```

The search.lookupFields(options) method also includes a promise version, `search.lookupFields.promise(options)`. For more information about promises, see Promise Object.

Note that the return contains either an object or a scalar value, depending on whether the looked-up field holds a single value, or a collection of values. Single select fields are returned as an object with value and text properties. Multi-select fields are returned as an object with value: text pairs.

In the following example, a select field like `my_select` would return an array of objects containing a value and text property. This select field contains multiple entries to select from, so each entry would have a numerical id (the value) and a text display (the text).

For "internalid" in this particular code snippet, the sample returns 1234. The internal id of a record is a single value, so a scalar is returned.

```
{
    internalid: 1234,
    firstname: 'Joe',
    my_select: [{
        value: 1,
        text: 'US Sub'
    }],
    my_multiselect: [{
        "value": "1,2",
        "text": "US Sub, EU Sub"
    }]
}
```

### Returns

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object</td>
<td>array</td>
</tr>
<tr>
<td>-</td>
<td>Returns select fields as an object with value and text properties.</td>
<td></td>
</tr>
<tr>
<td>-</td>
<td>Returns multiselect fields as an array of object with value: text pairs.</td>
<td></td>
</tr>
</tbody>
</table>

### Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

### Governance

1 unit

### Module

N/search Module

### Since

2015.2

---

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>enum</td>
<td>Required</td>
<td>The search type for which you want to look up fields. Use the search.Type enum for this argument.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.id</td>
<td>string</td>
<td>Required</td>
<td>Internal ID for the record, for example 777 or 87.</td>
<td>2015.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>--------------------</td>
<td>------------------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>options.columns</td>
<td>string</td>
<td>Optional</td>
<td>Array of column/field names to look up, or a single column/field name. The columns parameter can also be set to reference joined fields.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>(1): Missing a required argument: (2)</td>
<td>Required parameter is missing.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var fieldLookUp = search.lookupFields({
  type: search.Type.SALES_ORDER,
  id: '87',
  columns: ['entity', 'subsidiary', 'name', 'currency']
});
...
//Add additional code
```

**search.lookupFields.promise(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Performs a search asynchronously for one or more body fields on a record. Returns select fields as an object with value and text properties. Returns multiselect fields as an object with value:text pairs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>For information about the parameters and errors thrown for this method, see search.lookupFields(options). For additional information on promises, see Promise Object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>object</th>
<th>array</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Synchronous Version</th>
<th>search.lookupFields(options)</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>All client-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Client Script Type.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>1 unit</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/search Module</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>2015.2</th>
</tr>
</thead>
</table>
Syntax

```javascript
// Add additional code
...
search.lookupFields.promise({
    type: search.Type.EMPLOYEE,
    id: -5,
    columns : 'email'
})
.then(function (result) {
    log.debug({
        details: "Completed: " + result
    });
    // do something after completion
})
.catch(function onRejected(reason) {
    // do something on rejection
});
...
// Add additional code
```

### search.createColumn(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a new search column as a search.Column object.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Important:</strong></td>
<td>You cannot directly create a filter or column for a list/record type field in SuiteScript by passing in its text value. You must use the field's internal ID. If you must use the field's text value, you can create a filter or column with a formula using name: 'formulatext'.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>search.Column</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/search Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### 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>options.name</td>
<td>string</td>
<td>Required</td>
<td>Name of the search column. See Column.name.</td>
<td>2015.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td>Since</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------</td>
<td>---------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------</td>
</tr>
<tr>
<td>options.join</td>
<td>string</td>
<td>Optional</td>
<td>Join ID for the search column. See Column.join.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.summary</td>
<td>enum</td>
<td>Optional</td>
<td>Summary type for the column. See search.Summary and Column.summary.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.formula</td>
<td>string</td>
<td>Optional</td>
<td>Formula for the search column. See Column.formula.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.function</td>
<td>string</td>
<td>Optional</td>
<td>Special function for the search column. See Column.function.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>Optional</td>
<td>Label for the search column. See Column.label.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.sort</td>
<td>enum</td>
<td>Optional</td>
<td>The sort order of the column. Use the search.Sort enum for this argument.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Also see Column.sort.</td>
<td></td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>{1}: Missing a required argument: {2}</td>
<td>Required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_COLUMN_SUM</td>
<td>A search.Column object contains an invalid column summary type, or is not in proper syntax: {1}.</td>
<td>The options.summary parameter is not a valid search summary type. See search.Summary.</td>
</tr>
<tr>
<td>INVALID_SRCH_FUNCTN</td>
<td></td>
<td>An unknown function is provided.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var currencyColumn = search.createColumn({
  name: 'currency',
  sort: search.Sort.ASC
});
...
//Add additional code
```

```
search.createFilter(options)
```

**Method Description**

Creates a new search filter as a search.Filter object.
Important: You cannot directly create a filter or column for a list/record type field in SuiteScript by passing in its text value. You must use the field's internal ID. If you must use the field's text value, you can create a filter or column with a formula using name: 'formulatext'.

Returns

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Name or internal ID of the search field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.join</td>
<td>string</td>
<td>Optional</td>
<td>Join ID for the search filter.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.operator</td>
<td>search.Operator</td>
<td>Required</td>
<td>Operator used for the search filter. Use the search.Operator enum.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.values</td>
<td>string</td>
<td>Date</td>
<td>number</td>
<td>string[]</td>
</tr>
<tr>
<td>options.formula</td>
<td>string</td>
<td>Optional</td>
<td>Formula used by the search filter.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.summary</td>
<td>search.Summary</td>
<td>Optional</td>
<td>Summary type for the search filter. See search.Summary.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>(1): Missing a required argument: (2)</td>
<td>Required parameter is missing.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_FILTER_SUM</td>
<td>A search.Column object contains an invalid column summary type, or is not in proper syntax: (1).</td>
<td>options.summary parameter is not a valid search summary type. See search.Summary.</td>
</tr>
<tr>
<td>SSS_INVALID_SRCH_OPERATOR</td>
<td>An search.Filter object contains an invalid operator, or is not in proper syntax: (1).</td>
<td>options.operator parameter is not a valid operator type. See search.Operator.</td>
</tr>
</tbody>
</table>
Syntax

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
// Add additional code
...
// Create a filter joined to another record type. When you create a joined filter:
// - The name property is the field ID of the field in the joined record that you are filtering on
// - The join property is the field ID of the field in the current record that contains the record
// type you want to join to
// - The operator property is the operator to use to filter the results
// - The values property contains the values to use to filter the results
// For example, the following search definition lists the first 100 employees found
// who have a custom role. The search definition specifies that the search applies to
// Employee records. The filter definition joins the Role record type to the search
// and returns results where the iscustom field (a field on the Role record) is true.
var result = search.create(
  type: 'employee',
  columns: ['firstname', 'lastname', 'role'],
  filters: [
    search.createFilter(
      name: 'iscustom',
      join: 'role',
      operator: search.Operator.IS,
      values: true
    )
  ]
).run().getRange({
  start: 0,
  end: 100
});
log.debug({
  title: 'Result',
  details: result
});
...
// Add additional code
```

search.Operator

**Enum Description**

Enumeration that holds the values for search operators to use with the search.Filter.

See the help topic SuiteScript 1.0 Search Operators for more information about the field types supported for each operator type.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>All script types</th>
</tr>
</thead>
</table>
For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Module</th>
<th>N/search Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Values

- AFTER
- ALLOF
- ANY
- ANYOF
- BEFORE
- BETWEEN
- CONTAINS
- DOESNOTCONTAIN
- DOESNOTSTARTWITH
- EQUALTO
- GREATERTHAN
- GREATERTHANOREQUALTO
- HASKEYWORDS
- IS
- ISEMPTY
- ISNOT
- ISNOTEMPTY
- LESSTHAN
- LESSTHANOREQUALTO
- NONEOF
- NOTAFTER
- NOTALLOF
- NOTBEFORE
- NOTBETWEEN
- NOTEQUALTO
- NOTEQUALTO
- NOTGREATERTHAN
- NOTGREATERTHANOREQUALTO
- NOTGREATERTHANOREQUALTO
- NOTLESSTHAN
- NOTLESSTHANOREQUALTO
- NOTON
- NOTONORAFTE
- NOTONORBEFORE
- NOTWITHIN
- ON
- ONORAFTE
- ONORBEFORE
- STARTSWITH
- WITHIN

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code...
var mySearchFilter = search.createFilter({
    name: 'entity',
    operator: search.Operator.ISEMPTY
});
...

//Add additional code
```

### search.Sort

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the values for supported sorting directions used with search.createColumn(options).</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>All script types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

| Module | N/search Module          |
# search Module

## Since

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Values

- ASC
- DESC
- NONE

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/search Module Script Samples.

```javascript
//Add additional code
...
var currencyColumn = search.createColumn({
  name: 'currency',
  sort: search.Sort.ASC
});
...
//Add additional code
```

## search.Summary

**Enum Description**

Enumeration that holds the values for summary types used by the Column.summary or Filter.summary properties. For more information about each summary type, see the help topic SuiteScript 1.0 Search Summary Types.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/search Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Values

- GROUP
- COUNT
- SUM
- AVG
- MIN
- MAX
Syntax

```
//Add additional code
...
var mySearchFilter = search.createFilter({
  name: 'entity',
  summary: search.Summary.GROUP
});
...
//Add additional code
```

`search.Type`

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for search types supported in the N/search Module. This enum is used to pass the type argument to <code>search.create(options)</code>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/search Module

**Since**

2015.2

**Values**

```
ACCOUNT
ACCOUNTING_BOOK
ACCOUNTING_CONTEXT
ACCOUNTING_PERIOD
ACTIVITY
ADV_##INTER_##COMPANY_##JOURNAL_##ENTRY
AMORTIZATION_##SCHEDULE
AMORTIZATION_##TEMPLATE
ASSEMBLY_BUILD
ASSEMBLY_ITEM
ASSEMBLY_UNBUILD
BILLING_ACCOUNT
BILLING_##ACCOUNT_##BILL_##CYCMECER_##COMPANY_##TRANSFER_##ORDER
BILLING_##ACCOUNT_##BILL_##REQUEST
FIN_RPT_AGGREGATE_F_R
FIXED_##AMOUNT_##PROJECT_##REVENUE
FOLDER
FULFILLMENT_REQUEST
GENERIC_RESOURCE
GIFT_CERTIFICATE
GLOBAL_##ACCOUNT_##MAPPING
GLOBAL_##INVENTORY_##RELATIONSHIP
GL_LINES_AUDIT_LOG
INBOUND_SHIPMENT
INTER_##COMPANY_##JOURNAL_##ENTRY
INTER_##COMPANY_##TRANSFER_##ORDER
INTER_##COMPANY_##TRANSFER_##REQUEST
INTER_##COMPANY_##TRANSFER_##REVERSAL
INTER_##COMPANY_##TRANSFER_##REVERSAL
INTER_##COMPANY_##TRANSFER_##REVERSAL
PROJECT_TEMPLATE
REVENUE_ALARM
REVENUE_ALLOCATION
REVENUE_##COMMITMENT_##REVERSAL
REVENUE_##COMMITMENT_##REVERSAL
RES_##ALLOCATION_##TIME_##OFF_##CONFLICT
RESTLET
RETURN_AUTHORIZATION
REVENUE_ARRANGEMENT
```
<table>
<thead>
<tr>
<th>N/search Module</th>
<th>INVENTORY_BALANCE</th>
<th>REVENUE_PLAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>BILLING_CLASS</td>
<td>INVENTORY_COUNT</td>
<td>REV_REC_SCHEDULE</td>
</tr>
<tr>
<td>BILLING_RATE_CARD</td>
<td>INVENTORY_DETAIL</td>
<td>REV_REC_TEMPLATE</td>
</tr>
<tr>
<td>BILLING_REVENUE_EVENT</td>
<td>INVENTORY_ITEM</td>
<td>ROLE</td>
</tr>
<tr>
<td>BILLING_SCHEDULE</td>
<td>INVENTORY_NUMBER</td>
<td>SALES_ORDER</td>
</tr>
<tr>
<td>BIN</td>
<td>INVENTORY_STATUS</td>
<td>SALES_ROLE</td>
</tr>
<tr>
<td>BIN_TRANSFER</td>
<td>INVOICE</td>
<td>SALES_TAX_ITEM</td>
</tr>
<tr>
<td>BIN_WORKSHEET</td>
<td>INVOICE_TRANSFER</td>
<td>SAVED_SEARCH</td>
</tr>
<tr>
<td>BLANKET_PURCHASE_ORDER</td>
<td>INVENTORY_STATUS_STATUS_CHANGE</td>
<td>SCHEDULED_SCRIPT</td>
</tr>
<tr>
<td>BOM</td>
<td></td>
<td>SCHEDULED_SCRIPT_INSTANCE</td>
</tr>
<tr>
<td>BOM_REVISION</td>
<td></td>
<td>SCRIPT_DEPLOYMENT</td>
</tr>
<tr>
<td>BUNDLE_INSTALLATION_SCRIPT_ISSUE</td>
<td>ITEM</td>
<td>SERIALIZED.Assembly.Item</td>
</tr>
<tr>
<td>CALENDAR_EVENT</td>
<td>ITEM_ACCOUNT_MAPPING</td>
<td>SERIALIZED.Inventory.Item</td>
</tr>
<tr>
<td>CAMPAIGN</td>
<td>ITEM_BIN_NUMBER</td>
<td>SERVICE_ITEM</td>
</tr>
<tr>
<td>CASH_REFUND</td>
<td>ITEM_DEMAND_PLAN</td>
<td>SHIP_ITEM</td>
</tr>
<tr>
<td>CASH_SALE</td>
<td>ITEM_FULFILLMENT</td>
<td>SOLUTION</td>
</tr>
<tr>
<td>CHARGE</td>
<td>ITEM_GROUP</td>
<td>STATISTICAL_JOURNAL_ENTRY</td>
</tr>
<tr>
<td>CHARGE_RULE</td>
<td>ITEM_RECEIPT</td>
<td>STORE_PICKUP_FULFILLMENT</td>
</tr>
<tr>
<td>CHECK</td>
<td>ITEM_REVISION</td>
<td>SUBSCRIPTION</td>
</tr>
<tr>
<td>CLASSIFICATION</td>
<td>ITEM_SUPPLY_PLAN</td>
<td>SUBSCRIPTION_INTERFACE</td>
</tr>
<tr>
<td>CLIENT_SCRIPT</td>
<td>JOB</td>
<td>SUBSCRIPTION_LINE</td>
</tr>
<tr>
<td>CMS_CONTENT</td>
<td>JOB_STATUS</td>
<td>SUBSCRIPTION_PLAN</td>
</tr>
<tr>
<td>CMS_CONTENT_TYPE</td>
<td></td>
<td>SUBSCRIPTION_RENEWAL_HISTORY</td>
</tr>
<tr>
<td>COM_SEARCH_GROUP_SYS JOB_TYPE</td>
<td></td>
<td>SUBSIDIARY</td>
</tr>
<tr>
<td>COM_SEARCH.ONE_WAYENARIO</td>
<td></td>
<td>SUBTOTAL_ITEM</td>
</tr>
<tr>
<td>COMMERCE_CATEGORY</td>
<td>KIT_ITEM</td>
<td>SUITE_SCRIPT_DETAIL</td>
</tr>
<tr>
<td>COMPETITOR</td>
<td>LABOR_BASED_PROJECT_REV_RECEIEN</td>
<td>SUPPLY_CHAIN_SNAPSHOT</td>
</tr>
<tr>
<td>CONSOLIDATED_EXCHANGE_RATE</td>
<td>LOCATION</td>
<td>SUPPORT_CASE</td>
</tr>
<tr>
<td>CONTACT</td>
<td>LOT_NUMBERED.Assembly.Assembly</td>
<td>SYSTEM_NOTE</td>
</tr>
<tr>
<td>CONTACT_CATEGORY</td>
<td>LOT_NUMBERED.Inventory.Inventory</td>
<td>SYSTEM_TASK</td>
</tr>
<tr>
<td>CONTACT_ROLE</td>
<td>MANUFACTURING_COSTTEMPLATE</td>
<td>TAX_DETAIL</td>
</tr>
<tr>
<td>COST_CATEGORY</td>
<td>MANUFACTURING_OPERATION_TAX</td>
<td>TAX_GROUP</td>
</tr>
<tr>
<td>COUPON_CODE</td>
<td>MAP_REDUCE_SCRIPT</td>
<td>TAX_PERIOD</td>
</tr>
<tr>
<td>CREDIT_CARD_CHANGE</td>
<td>MANUFACTURING_ROUTING</td>
<td>TAX_TYPE</td>
</tr>
<tr>
<td>CREDIT_CARD_REFUND</td>
<td>MAP_Reduce_script</td>
<td>TERM</td>
</tr>
<tr>
<td>CREDIT_MEMO</td>
<td>MARKUP_ITEM</td>
<td>TIME_APPROVAL</td>
</tr>
<tr>
<td>CURRENCY</td>
<td>MASSUPDATE_SCRIPT</td>
<td>TIME_BILL</td>
</tr>
<tr>
<td>CUSTOM</td>
<td>MERCHANDISE_HIERARCHY_LEVEL</td>
<td>VERSION</td>
</tr>
<tr>
<td>CUSTOMER_CATEGORY</td>
<td>MERCHANDISE_HIERARCHY_NODE_TIME_BILL</td>
<td>VERSION</td>
</tr>
<tr>
<td>CUSTOMER_DEPOSIT</td>
<td>MERCHANDISE_HIERARCHY_VERSION_TIME_OFF_CHANGE</td>
<td>VERSION</td>
</tr>
<tr>
<td>CUSTOMER_MESSAGE</td>
<td>MFG_PLANNED_TIME</td>
<td>VERSION</td>
</tr>
<tr>
<td>CUSTOMER_PAYMENT</td>
<td>MESSAGE</td>
<td>TIME_OFF_PLAN</td>
</tr>
<tr>
<td>CUSTOMER_PAYMENT AUTHORIZATION</td>
<td>MFG_Planned_Time</td>
<td>TIME_OFF_REQUEST</td>
</tr>
<tr>
<td>CUSTOMER_REFUND</td>
<td>NON_INVENTORY_ITEM</td>
<td>TIME_OFF_RULE</td>
</tr>
<tr>
<td>CUSTOMER_STATUS</td>
<td>NOTE</td>
<td>TIME_OFF_TYPE</td>
</tr>
<tr>
<td>CUSTOM_RECORD</td>
<td>NOTE_TYPE</td>
<td>TOPIC</td>
</tr>
<tr>
<td>CUSTOM_TRANSACTION</td>
<td>OPPORTUNITY</td>
<td>TRANSACTION</td>
</tr>
<tr>
<td>DELETED_RECORD</td>
<td>OTHER_CHARGE_ITEM</td>
<td>TRANSFER_ORDER</td>
</tr>
<tr>
<td></td>
<td></td>
<td>UBER</td>
</tr>
</tbody>
</table>
## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/search Module Script Samples](#).

```javascript
//Add additional code
...
var mySearch = search.create(
    type: search.Type.CUSTOMER,
    filters: filters,
    columns: columns
));
...
//Add additional code
```

## N/sftp Module

The sftp module provides a way to upload and download files from external SFTP servers.

SFTP servers can be hosted by your organization or by a third party. NetSuite does not provide SFTP server functionality.

All SFTP transfers to or from NetSuite must originate from SuiteScript. It is not possible for external clients to initiate file transfers using SFTP.
Note: To use an external server to initiate a NetSuite file transfer that doesn't use SFTP, you can use RESTlets or SuiteTalk (Web Services). In SuiteScript, RESTlets can respond to requests containing file data and save them in the File Cabinet. RESTlets can also respond to requests for file data by loading the contents from the File Cabinet and returning them in the response. Note that binary file content must be received or sent as Base64 encoded Strings. See the help topic SuiteScript 2.0 RESTlet Script Type for more information.

In SuiteTalk, applications can invoke CRUD operations on the File Record to populate or change the contents of the File Cabinet. See the help topics SuiteTalk (Web Services) Platform Guide and File for more information.

- N/sftp Module Members
- Connection Object Members
- N/sftp Module Script Sample
- Setting up an SFTP Transfer
- SFTP Authentication
- Supported Cipher Suites and Host Key Types
- Supported SuiteScript File Types

### N/sftp Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>sftp.Connection</td>
<td>Object</td>
<td>Server-side scripts</td>
<td>Represents a connection to the account on the remote FTP server.</td>
</tr>
<tr>
<td>Method</td>
<td>sftp.###create###Connection(options)</td>
<td>sftp.Connection</td>
<td>Server-side scripts</td>
<td>Establishes a connection to a remote FTP server.</td>
</tr>
</tbody>
</table>

### Connection Object Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Connection.###download###(options)</td>
<td>Server-side scripts</td>
<td></td>
<td>Downloads a file from the remote FTP server</td>
</tr>
<tr>
<td></td>
<td>Connection.###upload###(options)</td>
<td>Server-side scripts</td>
<td></td>
<td>Uploads a file to the remote FTP server.</td>
</tr>
</tbody>
</table>
N/sftp Module Script Sample

**Note:** This sample script uses the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

**Important:** Before you run this script, you must replace the GUID and host key with one specific to your account. The user name, URL, and directory values in this sample are also placeholders. Before using this sample, replace the placeholder values with valid values from your NetSuite account.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

The following example uploads and downloads a file.

To obtain a real host key, use `ssh-keyscan <domain>`.

To create a real password GUID, obtain a password value from a credential field on a form. For more information, see `Form.addCredentialField(options)`. Also see N/https Module Script Sample for a Suitelet example that shows creating a form field that generates a GUID.

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

  var myPwdGuid = "B34672495064525E5D65032D63B52301";
  var myHostKey = "AAA1234567890Q=";

  // establish connection to remote FTP server
  var connection = sftp.createConnection({
    username: 'myuser',
    passwordGuid: myPwdGuid, // references var myPwdGuid
    url: 'host.somewhere.com',
    directory: 'myuser/wheres/my/file',
    hostKey: myHostKey // references var myHostKey
  });

  // specify the file to upload using the N/file module
  var myFileToUpload = file.create({
    name: 'originalname.js',
    fileType: file.fileType.PLAINTEXT,
    contents: 'I am a test file. Hear me roar.'
  });

  // upload the file to the remote server
  connection.upload({
    directory: 'relative/path/to/remote/dir',
    filename: 'newFileNameOnServer.js',
    file: myFileToUpload,
    replaceExisting: true
  });
});
```
Setting up an SFTP Transfer

- Development Preparation for SFTP transfers
- Execution of an SFTP transfer

Development Preparation for SFTP transfers

To successfully connect to your SFTP server with SuiteScript, the following steps are recommended:

1. Talk to your SFTP service provider about your plans.
   - Determine the connection properties required to connect with your external SFTP server. For example:
     - username
     - password
     - url
     - port
     - upload/download directories
     - host key
     - host key type
   - Make sure that you know your provider's practices around host key changes, maintenance and failover. For example, find out if there are multiple URLs or ports to try.
   - Check compatibility with the SFTP ciphers supported by NetSuite. See Supported Cipher Suites and Host Key Types.
   - Determine if your provider requires at-rest file encryption (in addition to what the SFTP protocol provides during transfer). Decide if you need to add file encryption.

2. Build a credential management Suitelet to capture username and password token. Then, test the connection.
   - Create custom fields to store the user's SFTP username and password token
   - Implement the Suitelet.
     a. Draw a form on a GET request.
     b. Save the username and password token on a POST request.
     c. Test the connection.
   - See Creating a Suitelet Form that Contains a Credential Field.
   - Build a server-side script to handle operations such as:
     - Load a File Cabinet file and upload it to the SFTP server.
     - Download an on demand file from the SFTP server and save it in File Cabinet.

```javascript
// download the file from the remote server

var downloadedFile = connection.download({
  directory: 'relative/path/to/file',
  filename: 'downloadMe.js'
});
```
Execution of an SFTP transfer

The following steps occur during a successful SFTP transfer using SuiteScript:

1. User submits their SFTP credentials via a Suitelet.
2. Suitelet captures and stores the credential token.
3. A server-side script is triggered.
4. Script identifies the appropriate credential token and other connection attributes, and establishes the SFTP connection.
5. Script requests the transfer.

SFTP Authentication

Please review the following sections for an overview of SFTP authentication when using SuiteScript.

- Credential Tokenization
- Creating a Suitelet Form that Contains a Credential Field
- Reading the Credential Token in a Suitelet
- Credential Management
- Credential GUID Persistence
- Protocols
- Host Key Verification
- Retrieving the Host Key of an External SFTP Server

Only username/password based authentication is supported. Public key based authentication is not supported.

Credential Tokenization

SuiteScript provides the ability for users to securely store authentication credentials in such a way that scripts are able to utilize encrypted saved credentials without being able to see their contents. The script author must specify which scripts and domains are permitted for use with the credential. To restrict the credential for use by SuiteScript automation triggered by the same user who originally saved the credential, the script author can set the `restrictToCurrentUser` parameter.

Creating a Suitelet Form that Contains a Credential Field

```javascript
// Note: Credential fields have a default maximum length of 32 characters. If needed, use the Field.maxLength property to change this value

if(request.method === context.Method.GET){
    var form = ui.createForm({title: 'Enter SFTP Credentials'});
    var credField = form.addCredentialField({
        id: 'custfield_sftp_password_token',
        label: 'SFTP Password',
        restrictToScriptIds: ['customscript_sftp_script'],
        restrictToDomains: ['acmebank.com'],
        restrictToCurrentUser: true //Depends on use case
    });
    credField.maxLength = 64;
    form.addSubmitButton();
}
```
Reading the Credential Token in a Suitelet

Note that the following code snippet is not a fully functional sample.

```javascript
... 
var request = context.request;
if(request.method === context.Method.POST){
    // Read the request parameter matching the field ID we specified in the form
    var passwordToken = request.parameters.custfield_sftp_password_token;
    log.debug({
        title: 'New password token',
        details: passwordToken
    });
    // In a real-world script, "passwordToken" is saved into a custom field here...
}
... 
```

Credential Management

User passwords can be stored using secure Credential Fields. This type of field is available on the serverWidget.Form Object in the N/ui/serverWidget Module.

Encrypted custom fields do not support tokenization and are not compatible with the SFTP module. Instead, you can add a credential field using Form.addCredentialField(options).

Credential GUID Persistence

Scripts may store credential tokens as convenient for the script author. Credential tokens are not related to the password in its original or encrypted form within NetSuite. These tokens are unique identifiers which allow a script to refer to an encrypted secret securely stored within the SuiteCloud platform. Automatic password expiration is not currently provided, nor is it possible to view an inventory of saved credentials in the user interface.

Protocols

The SFTP module allows scripts to transfer files using the SSH File Transfer Protocol only. Other file-based protocols such as FTP, FTPS, SCP are not supported by this module.

Host Key Verification

An SFTP server identifies itself using a host key when a client attempts to establish a connection. Host keys are unique keys that the underlying SSH protocol uses to allow the server to provide a fingerprint. Clients can verify that the expected server has responded to the connection request for a particular URL and port number.

SuiteScript requires that the host key is provided by the script attempting to connect so that the SFTP module can check the identity of the SFTP server. This security best practice is commonly referred to as "Strict Host Key Checking".

Host keys are used to verify the identity of the server, not the client. SFTP/SSH host keys have no relationship to SFTP key based authentication, which is not currently supported.
By design, there is no SuiteScript API call for checking the host key of a remote SFTP server, or an option to disable strict host key checking. The script must always know the host key ahead of time.

We recommend using OpenSSH’s ssh-keyscan tool to check the host key of an external SFTP site. See Retrieving the Host Key of an External SFTP Server and The OpenBSD’s ssh-keyscan page.

Retrieving the Host Key of an External SFTP Server

An example usage checking the RSA host key of URL: acme.com at port: 1234 from a *nix shell follows:

```
$ ssh-keyscan -t rsa -p 1234 acme.com
AATpn1P9jBe0q9h9ue741245x75x9c1kh60v5xvsn=
```

It is recommended to always pass the key type and port number. This practice helps to avoid ambiguity in the response from the external SFTP server.

Supported Cipher Suites and Host Key Types

SFTP connections are encrypted. For security reasons, NetSuite requires that the server to which a connection request is being made supports at least one of the following ciphers aes128-ctr, aes192-ctr or aes256-ctr. The preceding cipher specs refer to the AES cipher in Counter stream cipher mode using 128, 192 or 256 bit key sizes.

To check interoperability of your SFTP server or service provider, refer to the following table:

<table>
<thead>
<tr>
<th>Communication protocol</th>
<th>SFTP (SSH + FTP) is supported.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Only CTR (and not CBC) ciphers are allowed. Your SFTP server can use the following encryption algorithms:</td>
</tr>
<tr>
<td></td>
<td>- AES 128-CTR</td>
</tr>
<tr>
<td></td>
<td>- AES 192-CTR</td>
</tr>
<tr>
<td></td>
<td>- AES 256-CTR</td>
</tr>
<tr>
<td></td>
<td>Files are not additionally encrypted during transfer. The entire transmission is encrypted by the SSH protocol.</td>
</tr>
<tr>
<td>Authentication mechanism</td>
<td>Username/password only (not key-based)</td>
</tr>
<tr>
<td>SSH host key</td>
<td>With each connection request, you must supply the host key. Any host key changes need to be managed manually.</td>
</tr>
<tr>
<td>Guid</td>
<td>The password guid should be a value generated by a credential field from a Suitelet using Form.addCredentialField(options).</td>
</tr>
<tr>
<td></td>
<td>The password guid field's originating credential field must include the SFTP domain on the restrictToDomains parameter.</td>
</tr>
<tr>
<td></td>
<td>The password guid field's originating credential field must include the script utilizing the password guid on the restrictToScriptIds parameter.</td>
</tr>
</tbody>
</table>

Firewall policy is at the discretion of your SFTP service provider.

Supported SuiteScript File Types

SuiteScript has two types of file objects: previously existing files in the NetSuite File Cabinet, and on demand files created using SuiteScript API calls such as file.create(options) or Connection.download(options).
File Cabinet and on demand files are supported by `Connection.upload(options)`.

Note that `Connection.download(options)` returns an on demand file object. For an on demand file to be saved into the File Cabinet, it must receive a folder id and be explicitly saved.

```javascript
... var downloadedFile = sftp.download({...});
downloadedFile.folder = 1234;
downloadedFile.save();
... 
```

⚠️ **Important:** It’s possible that a file you are downloading may be encrypted, or your SFTP provider may expect an uploaded file in a encrypted format in accordance with that provider’s security practices. Make sure that you understand your provider’s expectations and the cryptographic capabilities in SuiteScript (see [N/crypto Module](#)).

### Annotated Syntax Sample

```javascript
require(['N/sftp', 'N/file'],
        function (sftp, file)
        {
            var connection = sftp.createConnection({
                username: 'myuser',
                passwordGuid: 'B34672495064525E5D65032D63B52301',
                url: 'host.somewhere.com',
                port: 22,
                directory: 'transferfiles',
                hostKey: require('N/crypto').getHostKey()
            });

            var file = file.createFile({
                file: 'newfile.txt',
                size: 100, // optional
                type: file.FILE_TYPE_REGULAR // optional
            });

```

Annotate the code with comments to explain the purpose and logic of each part. This will help in understanding the code and improving its maintainability. For example:

- **Username supplied by the administrator of the external SFTP server.**
- **Password Token/GUID obtained by reading the form POST parameter associated with user submission of a form containing a Credential Field.**
- **URL supplied by the administrator of the external SFTP server.**
- **SFTP Port number supplied by the administrator of the external SFTP server (defaults to 22).**
- **Transfer directory supplied by the administrator of the external SFTP server (optional).**
- **RSA Host Key obtained via ssh-keyscan tool.**

```javascript

```

Refer to the [SuiteScript API](#) for more information on the `Connection` object and its methods. The `createConnection` method is used to initiate a SFTP connection, where parameters such as `username`, `passwordGuid`, `url`, `port`, and `directory` are provided. The `createFile` method is then used to create a new file, specifying its name, size, and type, if necessary. The `hostKey` is obtained using the `getHostKey` method from the `N/crypto` module, which is necessary for verifying the authenticity of the SFTP server.
hostKey: "AATpm1P9jB+cQx93q9UeZjA1245X78BDcRlKh+8ok56VzSw=="
});

/*
Creating a simple file.
*/
var myFileToUpload = file.create({
    name: 'originalname.js',
    fileType: file.fileType.PLAINTEXT,
    contents: 'I am a test file. Hear me roar.'
});

/*
Uploading the file to the external SFTP server.
*/
connection.upload({
    /*
    Subdirectory within the transfer directory specified when connecting (optional).
    */
    directory: 'relative/path/to/remote/dir',
    /*
    Alternate file name to use instead of the one given to the file object (optional).
    */
    filename: 'newFileNameOnServer.js',
    /*
    The file to upload.
    */
    file: myFileToUpload,
    /*
    If a file already exists with that name, replace it instead of failing the upload.
    */
    replaceExisting: true
});

var downloadedFile = connection.download({
    /*
    Subdirectory within the transfer directory specified when connecting (optional).
    */
    directory: 'relative/path/to/file',
    /*
    The name of the file within the above directory on the external SFTP server which to download.
    */
    filename: 'downloadMe.js'
});

});

sftp.Connection

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Represents a connection to the account on the remote FTP server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/sftp Module Script Sample.

```javascript
//Add additional code ...
// establish connection to the FTP server
var objConnection = sftp.createConnection({
    username: 'username',
    passwordGuid: pwdGuid,
    url: 'host.somewhere.com',
    directory: 'username/where/you/file'
});
...
//Add additional code
```

### Connection.download(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Downloads a file from the remote FTP server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>file.File Object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>100</td>
</tr>
<tr>
<td>Module</td>
<td>N/sftp Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

#### Parameters

⚠️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.filename</td>
<td>string</td>
<td>Required</td>
<td>The name of the file to download.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.directory</td>
<td>string</td>
<td>Optional</td>
<td>The relative path to the directory that contains the file to download.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, the path is set to the current directory.</td>
<td></td>
</tr>
<tr>
<td>options.timeout</td>
<td>number</td>
<td>Optional</td>
<td>The number of seconds to allow for the file to download.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

⚠️ **Important:** This input must take the form of a relative path.
### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP_MAXIMUM_FILE_SIZE_EXCEEDED</td>
<td>The file size is greater than the maximum file size allowed by NetSuite.</td>
</tr>
<tr>
<td>FTP_INVALID_DIRECTORY</td>
<td>The directory does not exist on the remote FTP server.</td>
</tr>
<tr>
<td>FTP_TRANSFER_TIMEOUT_EXCEEDED</td>
<td>The transfer is taking longer than the specified options.timeout value.</td>
</tr>
<tr>
<td>FTP_INVALID_TRANSFER_TIMEOUT</td>
<td>The options.timeout value is either a negative value, zero or greater than 300 seconds.</td>
</tr>
<tr>
<td>FTP_FILE_DOES_NOT_EXIST</td>
<td>The options.filename does not exist in the options.directory location.</td>
</tr>
<tr>
<td>FTP_PERMISSION_DENIED</td>
<td>Access to the file or directory on the remote FTP server was denied.</td>
</tr>
<tr>
<td>CONNECTION_RESET</td>
<td>The connection was reset.</td>
</tr>
<tr>
<td>THE_REMOTE_PATH_FOR_FILE_IS_NOT_VALID</td>
<td>The file's remote path is invalid.</td>
</tr>
<tr>
<td>CONNECTION_CLOSED_BY_HOST</td>
<td>The connection was closed by the host.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/sftp Module Script Sample](#).

```javascript
//Add additional code
...
var downloadedFile = objConnection.download({
  directory: 'relative/path/to/file',
  filename: 'downloadMe.js'
});
...
//Add additional code
```

### Connection.upload(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Uploads a file to the remote FTP server.</td>
</tr>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server-side scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>100</td>
</tr>
<tr>
<td>Module</td>
<td>N/sftp Module</td>
</tr>
</tbody>
</table>
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.file</td>
<td>file.File</td>
<td>Required</td>
<td>The file to upload.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.filename</td>
<td>string</td>
<td>Optional</td>
<td>The name to give the uploaded file on server. By default, the filename is the same specified by options.file.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Note: Illegal characters are automatically escaped.</td>
<td></td>
</tr>
<tr>
<td>options.directory</td>
<td>string</td>
<td>Optional</td>
<td>The relative path to the directory where the file should be upload to.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, the path is set to the current directory.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Important: This input must take the form of a relative path.</td>
<td></td>
</tr>
<tr>
<td>options.timeout</td>
<td>number</td>
<td>Optional</td>
<td>The number of seconds to allow for the file to upload.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, this value is set to 300 seconds.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Important: This parameter does not specify the overall timeout limit. The value is only applied when no data is received within the specified period.</td>
<td></td>
</tr>
<tr>
<td>options.replaceExisting</td>
<td>boolean</td>
<td>Optional</td>
<td>Indicates whether the file being uploaded should overwrite any file with the name options.filename that already exists in options.directory.</td>
<td>2016.2</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td></td>
<td>If false, the FTP_FILE_ALREADY_EXISTS exception is thrown when a file with the same name already exists in the options.directory.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>false</td>
<td></td>
<td>By default, this value is false.</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP_INVALID_DIRECTORY</td>
<td>The directory does not exist on the remote FTP server.</td>
</tr>
</tbody>
</table>
Error Code | Thrown If
---|---
FTP_TRANSFER_TIMEOUT_EXCEEDED | The transfer is taking longer than the specified options.timeout value.
FTP_INVALID_TRANSFER_TIMEOUT | The options.timeout value is either a negative value, zero or greater than 300 seconds.
FTP_FILE_ALREADY_EXISTS | The options.replaceExisting value is false and a file with the same name exists in the remote directory.
CONNECTION_RESET | The connection was reset.
THE_REMOTE_PATH_FOR_FILE_IS_NOT_VALID | The file's remote path is invalid.
CONNECTION_CLOSED_BY_HOST | The connection was closed by the host.
FTP_PERMISSION_DENIED | Access to the file or directory on the remote FTP server was denied.

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/sftp Module Script Sample.

```javascript
//Add additional code ...
objConnection.upload({
  directory: 'relative/path/to/remote/dir',
  filename: 'newFileNameOnServer',
  file: myFileToUpload,
  replaceExisting: true
});
...
//Add additional code
```

sftp.createConnection(options)

**Method Description**

Establishes a connection to a remote FTP server.

To generate the passwordguid, you can create a suitelet that uses `Form.addCredentialField(options)`.

Use the N/https Module to fetch the GUID value returned from the Suitelet's credential field.

For more information, see Setting up an SFTP Transfer and Supported Cipher Suites and Host Key Types.

**Returns**
sftp.Connection, representing that connection.

**Supported Script Types**
All server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/sftp Module
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>Required</td>
<td>The host of the remote account.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.passwordGuid</td>
<td>string</td>
<td>Required</td>
<td>The password GUID for the remote account.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.hostKey</td>
<td>string</td>
<td>Required</td>
<td>The host key for the trusted fingerprint on the server.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.username</td>
<td>string</td>
<td>Optional</td>
<td>The username of the remote account. By default, the login is anonymous.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.port</td>
<td>number</td>
<td>Optional</td>
<td>The port used to connect to the remote account. By default, port 22 is used.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.directory</td>
<td>string</td>
<td>Optional</td>
<td>The remote directory of the connection. Note: The directory property is required if you use a remote server cannot resolve relative paths.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.timeout</td>
<td>number</td>
<td>Optional</td>
<td>The number of seconds to allow for an established connection. By default, this value is set to 20 seconds.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>
| options.hostKeyType | string  | Optional            | The type of host key specified by options.hostKey.

This value can be set to one of the following options:
- dsa
- ecdsa
- rsa

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTP_UNKNOWN_HOST</td>
<td>The host could not be found.</td>
</tr>
<tr>
<td>FTP_CONNECT_TIMEOUT_EXCEEDED</td>
<td>A connection could not be established within options.timeout seconds.</td>
</tr>
<tr>
<td>FTP_CANNOT_ESTABLISH_CONNECTION</td>
<td>The password/username was invalid or permission to access the directory was denied.</td>
</tr>
</tbody>
</table>
Error Code | Thrown If
--- | ---
FTP_INVALID_PORT_NUMBER | The port number is invalid.
FTP_INVALID_CONNECTION_TIMEOUT | The options.timeout value is either a negative value, zero, or greater than 20 seconds.
FTP_INVALID_DIRECTORY | The directory does not exist on the remote FTP server.
FTP_INVALID_DIRECTORY | The directory does not exist on the remote FTP server.
FTP_INVALID_DIRECTORY | The directory does not exist on the remote FTP server.
FTP_INCORRECT_HOST_KEY | The host key does not match the presented host key on the remote FTP server.
FTP_INCORRECT_HOST_KEY_TYPE | The host key type and provided host key type do not match.
FTP_MALFORMED_HOST_KEY | The host key is not in the correct format. (e.g. base 64, 96+ bytes)
FTP_PERMISSION_DENIED | Access to the file or directory on the remote FTP server was denied.
FTP_UNSUPPORTED_ENCRYPTION_ALGORITHM | The remote FTP server does not support one of NetSuite's approved algorithms. (e.g. aes256-ctr, es192-ctr, es128-ctr)
AUTHENTICATION_FAIL_TOO_MANY_INCORRECT | AUTHENTICATION_FAIL_TOO_MANY_INCORRECT
NO_ROUTE_TO_HOST_FOUND | No route to the host can be found.
CONNECTION_RESET | The connect was reset.
CONNECTION_CLOSED_BY_HOST | The connection was closed by the host.
THE_REMOTE_PATH_FOR_FILE_IS_NOT_VALID | The file's remote path is invalid.
SFTPCREDENTIAL_ENCODING_ERROR | There is an SFTP credential encoding error.
UNABLE_TO_GET_SFTP_SERVER_ADDRESS | The SFTP server address is unavailable.

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/sftp Module Script Sample.

```javascript
//Add additional code
...

// establish connection to the ftp server

var objConnection = sftp.createConnection({
  username: 'username',
  passwordGuid: pwdGuid, // references var pwdGuid
  url: 'host.somewhere.com',
  directory: 'username/wheres/my/file'
  hostKey: myHostKey // references var myHostKey
});
...

//Add additional code
```
N/sso Module

Use the sso module to generate outbound single sign-on (SuiteSignOn) tokens. For example, to create a reference to a SuiteSignOn record, or to integrate with an external application.

For more information about the SuiteSignOn feature, see the help topic Outbound Single Sign-on (SuiteSignOn).

- N/sso Module Member
- N/sso Module Script Sample

N/sso Module Member

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>sso.generateSuiteSignOnToken(options)</td>
<td>string</td>
<td>Portlet scripts, user event scripts, and Suitelets</td>
<td>Generates a new SuiteSignOn token for a user</td>
</tr>
</tbody>
</table>

N/sso Module Script Sample

The following sample script shows how you can use the sso module.

These sample scripts use the `require` function so that you can copy it into the debugger and test it. Keep in mind that you must use the `define` function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

**Important:** Before you run this script, you must replace the ID for the SuiteSignOn record with a value specific to your account. Additionally, the SuiteSignOn record you reference must be associated with a specific script. You make this association in the SuiteSignOn record's Connection Points sublist. For help with SuiteSignOn records, see the help topic Creating SuiteSignOn Records.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

The following example generates a new OAuth token for a user. The SuiteSignOn feature must be enabled.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/sso'],
    function(sso) {
        function generateSSOToken() {
            var suiteSignOnRecordId = 1;
            var url = sso.generateSuiteSignOnToken(suiteSignOnRecordId);
        }
        generateSSOToken();
    });

sso.generateSuiteSignOnToken(options)
```

Method Description | Method used to generate a new SuiteSignOn token for a user.
**Note:** To use this method, Outbound Single Sign-on and web services must be enabled in your account. To enable these features, go to Setup > Company > Enable Features. On the SuiteCloud tab, in the Manage Authentication section, select the SuiteSignOn check box. In the SuiteTalk section, select the Web Services check box. Click Save.

Returns
URL, OAuth token, and any integration variables as a string

**Supported Script Types**
Portlet scripts, user event scripts, and Suitelets
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
20 units

**Module**
N/sso Module

**Since**
2015.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.suiteSignOnId</td>
<td>string</td>
<td>required</td>
<td>The scriptId specified on the SuiteSignOn record.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To see a list of IDs for SuiteSignOn records, go to the SuiteSignOn list page (Setup &gt; Integration &gt; SuiteSignOn Setup &gt; Integration &gt; SuiteSignOn).</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** NetSuite recommends that you create a custom scriptId for each SuiteSignOn record to avoid naming conflicts should you decide use SuiteBundler to deploy your scripts into other accounts.

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_SSO</td>
<td>Invalid SuiteSignOn reference: (1). That SuiteSignOn object does not exist or has been marked as inactive.</td>
<td>The suiteSignOnId input parameter is invalid or does not exist.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The suiteSignOnId input parameter must be a scriptId and not a internal id.

| SSO_CONFIG_REQD | The SuiteSignOn object (1) is not configured for use with this script. You must specify the script as a connection point for this SuiteSignOn. | The suiteSignOnId input parameter is missing.                          |        |
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/sso Module Script Sample.

```javascript
//Add additional code
...
var suiteSignOnRecordId = 1;
var url = sso.generateSuiteSignOnToken('customsso1');
...
//Add additional code
```

Examples

This sample shows how to use `generateSuiteSignOnToken(options)` in a portlet script.

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType portlet
 * @NScriptPortletType form
 */

define(['N/sso'],
function (sso) {

  function render(context) {
    var suiteSignOnRecordId = 'customsso_test';
    var url = sso.generateSuiteSignOnToken(suiteSignOnRecordId);
    log.debug(url);
  }

  return {
    render: render
  };
})
```

This sample shows how to use `generateSuiteSignOnToken(options)` in a Suitelet script.

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType suitelet
 */

define(['N/sso'],
function (sso) {

  function onRequest(context) {
    var suiteSignOnRecordId = 'customsso_test';
    var url = sso.generateSuiteSignOnToken(suiteSignOnRecordId);
    log.debug(url);
  }

  return {
    render: render
  };
})
```
This sample shows how to use `generateSuiteSignOnToken(options)` in a user event script.

```javascript
/**
 * @NApiVersion 2.0
 * @NScriptType UserEventScript
 * @NModuleScope SameAccount
 */
define(['N/sso'],
    function(sso) {

        function beforeLoad(context) {
            var suiteSignOnRecordId = 'customsso_test';
            var url = sso.generateSuiteSignOnToken(suiteSignOnRecordId);
            log.debug(url);
        }

        return {
            beforeLoad: beforeLoad
        };
    });
)

N/task Module

Load the task module to create tasks and place them in the internal NetSuite scheduling or task queue. Use the task module to schedule scripts, run Map/Reduce scripts, import CSV files, merge duplicate records, and execute asynchronous workflows.

Each task type has its own corresponding object types. Use the methods available to each object type to configure, submit, and monitor the tasks.

Note: Regardless of task type, tasks are always triggered asynchronously.

- N/task Module Members
- ScheduledScriptTask Object Members
- ScheduledScriptTaskStatus Object Members
- MapReduceScriptTask Object Members
- MapReduceScriptTaskStatus Object Members
- CsvImportTask Object Members
- CsvImportTaskStatus Object Members
- EntityDeduplicationTask Object Members
- EntityDeduplicationTaskStatus Object Members
- SearchTask Object Members
- SearchTaskStatus Object Members
- WorkflowTriggerTask Object Members
- WorkflowTriggerTaskStatus Object Members
- RecordActionTask (Beta) Object Members (Beta)
- RecordActionTaskStatus (Beta) Object Members (Beta)
**Note:** The Record Action Task is a beta feature in 19.1.

### N/task Module Script Samples

#### N/task Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>task.##Scheduled##Object##Task</td>
<td>Server scripts</td>
<td>Server scripts</td>
<td>Encapsulates all the properties of a scheduled script task in SuiteScript. Use this object to place a scheduled script deployment into the NetSuite scheduling queue.</td>
</tr>
<tr>
<td></td>
<td>task.##Scheduled##Object##Task##Status</td>
<td>Server scripts</td>
<td></td>
<td>Encapsulates the status of a scheduled script placed into the NetSuite scheduling queue.</td>
</tr>
<tr>
<td></td>
<td>task.##Map##Reduce##Object##Task</td>
<td>Server scripts</td>
<td>Server scripts</td>
<td>Encapsulates a map/reduce script deployment.</td>
</tr>
<tr>
<td></td>
<td>task.##Map##Reduce##Object##Task##Status</td>
<td>Server scripts</td>
<td>Server scripts</td>
<td>Encapsulates the status of a map/reduce script deployment that has been submitted for processing.</td>
</tr>
<tr>
<td></td>
<td>task.##CsvImportTask</td>
<td>Object</td>
<td>Server scripts</td>
<td>Encapsulates the properties of a CSV import task. Use the methods and properties for this object to submit a CSV import task into the task queue and asynchronously import record data into NetSuite.</td>
</tr>
<tr>
<td></td>
<td>task.##Csv##Import##Object##Status</td>
<td>Server scripts</td>
<td></td>
<td>Encapsulates the status of a CSV import task placed into the NetSuite scheduling queue.</td>
</tr>
<tr>
<td></td>
<td>task.##Entity##Deduplication##Task</td>
<td>Object</td>
<td>Server scripts</td>
<td>Encapsulates all the properties of a merge duplicate records task request. Use the methods and properties of this object to submit a merge duplicate record job task into the NetSuite task queue.</td>
</tr>
<tr>
<td></td>
<td>task.##Entity##Deduplication##Task##Status</td>
<td>Server scripts</td>
<td></td>
<td>Encapsulates the status of a merge duplicate record task placed into the NetSuite task queue.</td>
</tr>
<tr>
<td></td>
<td>task.##SearchTask</td>
<td>Object</td>
<td>Server scripts</td>
<td>Encapsulates the properties required to initiate an asynchronous search.</td>
</tr>
<tr>
<td></td>
<td>task.##Search##Task##Status</td>
<td>Server scripts</td>
<td></td>
<td>Encapsulates the status of an asynchronous search initiation task that is placed into the NetSuite task queue.</td>
</tr>
<tr>
<td></td>
<td>task.##Workflow##Object##Task</td>
<td>Object</td>
<td>Server scripts</td>
<td>Encapsulates all the properties required to asynchronously initiate a workflow. Use WorkflowTriggerTask to create a task that initiates an instance of a specific workflow.</td>
</tr>
<tr>
<td></td>
<td>task.##Workflow##Object##Task##Status</td>
<td>Server scripts</td>
<td></td>
<td>Encapsulates the status of an asynchronous workflow initiation</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>task.##Record##Action##Task (Beta)</td>
<td></td>
<td>Encapsulates the properties of a record action task. Use this object to place a record action task into the NetSuite scheduling queue.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>task.##Record##Action##Task##Status (Beta)</td>
<td></td>
<td>Encapsulates the status of a record action task in the NetSuite scheduling queue.</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td>task.##create##(options)</td>
<td>task.ScheduledScriptTask</td>
<td>Creates an object for a specific task type and returns the task object.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>task.##checkStatus##(options)</td>
<td>task.RecordActionTask (Beta)</td>
<td>Returns a task status object associated with a specific task ID.</td>
</tr>
</tbody>
</table>

**Enum**

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task.TaskType</td>
<td>Enumeration that holds the string values for the types of task objects, supported by the N/task Module, that you can create with task.create(options).</td>
</tr>
<tr>
<td>task.TaskStatus</td>
<td>Enumeration that holds the string values for the possible status of tasks created and submitted with the N/task Module.</td>
</tr>
<tr>
<td>task.##Master##Selection##Mode</td>
<td>Enumeration that holds the string values for supported master selection modes when merging duplicate records with task.##Entity##Deduplication##Task.</td>
</tr>
<tr>
<td>task.DedupeMode</td>
<td>Enumeration that holds the string values for available deduplication modes when merging duplicate records with task.##Entity##Deduplication##Task.</td>
</tr>
<tr>
<td>task.##Dedupe##Entity##Type</td>
<td>Enumeration that holds the string values for entity types for which you can merge duplicate records with task.##Entity##Deduplication##Task.</td>
</tr>
<tr>
<td>task.##Map##Reduce##Stage</td>
<td>Enumeration that holds the string values for the stages of a map/reduce script deployment, which is encapsulated by the task.MapReduceScriptTask object.</td>
</tr>
<tr>
<td>task.##Action##Condition (Beta)</td>
<td>Enumeration that holds the string values for the possible record action conditions.</td>
</tr>
</tbody>
</table>
## ScheduledScriptTask Object Members

The following members are called on `task.ScheduledScriptTask`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>ScheduledScriptTask.submit()</code></td>
<td>string</td>
<td>Server scripts</td>
<td>Directs NetSuite to place a scheduled script deployment into the NetSuite scheduling queue and returns a unique ID for the task.</td>
</tr>
<tr>
<td>Property</td>
<td><code>ScheduledScriptTask.scriptId</code></td>
<td>number</td>
<td>string</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td><code>ScheduledScriptTask.deploymentId</code></td>
<td>number</td>
<td>string</td>
<td>Server scripts</td>
</tr>
<tr>
<td>Property</td>
<td><code>ScheduledScriptTask.params</code></td>
<td>Object</td>
<td>Server scripts</td>
<td>Object with key/value pairs that override the static script parameter field values on the script deployment.</td>
</tr>
</tbody>
</table>

## ScheduledScriptTaskStatus Object Members

The following members are called on `task.ScheduledScriptTaskStatus`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td><code>ScheduledScriptTaskStatus.scriptId</code></td>
<td>read-only number</td>
<td>Server scripts</td>
<td>Internal ID for a script record associated with a specific <code>task.ScheduledScriptTask</code> object.</td>
</tr>
<tr>
<td></td>
<td><code>ScheduledScriptTaskStatus.deploymentId</code></td>
<td>read-only number</td>
<td>Server scripts</td>
<td>Internal ID for a script deployment record associated with a specific <code>task.ScheduledScriptTask</code> object.</td>
</tr>
</tbody>
</table>

## MapReduceScriptTask Object Members

The following members are called on `task.MapReduceScriptTask`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>MapReduceScriptTask.submit()</code></td>
<td>string</td>
<td>Server scripts</td>
<td>Submits a map/reduce script deployment for processing.</td>
</tr>
<tr>
<td>Property</td>
<td><code>MapReduceScriptTask.scriptId</code></td>
<td>number</td>
<td>string</td>
<td>Server scripts</td>
</tr>
</tbody>
</table>
|             | `MapReduceScriptTask.deploymentId`        | number | string   | Server scripts | Internal ID (as a number), or script ID (as a string), for the script
<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPercentageCompleted()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the current percentage complete for the current stage of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPendingMapCount()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of records or rows not yet processed by the map stage of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getTotalMapCount()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of records or row inputs to the map stage of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPendingMapSize()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of bytes not yet processed by the map stage, as a component of total size, of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPendingReduceCount()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of records or rows not yet processed by the reduce stage of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPendingReduceSize()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of bytes not yet processed by the reduce stage, as a component of total size, of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPendingOutputCount()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of records or row inputs to the reduce stage of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPendingOutputSize()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of bytes not yet processed by the reduce stage, as a component of total size, of a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getTotalOutputCount()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total number of records or rows not yet processed by a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getTotalOutputSize()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the total size in bytes of all key/value pairs written as output, as a component of total size, by a task.MapReduceScriptTask.</td>
</tr>
<tr>
<td></td>
<td>MapReduceScriptTaskStatus.getPercentageComplete()</td>
<td>number</td>
<td>Server scripts</td>
<td>Returns the current percentage complete for the current stage of a task.MapReduceScriptTask.</td>
</tr>
</tbody>
</table>

MapReduceScriptTaskStatus Object Members

The following members are called on the task.MapReduceScriptTaskStatus object.
# MapReduceScriptTask Status

## getCurrentTotalSize()

**Return Type / Value Type:** `number`

**Supported Script Types:** `Server scripts`

**Description:**

Returns the total size in bytes of all stored work in progress by a task.MapReduceScriptTask.

## scriptId

**Read-only Number / String**

**Supported Script Types:** `Server scripts`

**Description:**

Internal ID for a map/reduce script record associated with a specific task.MapReduceScriptTask.

## deploymentId

**Read-only Number / String**

**Supported Script Types:** `Server scripts`

**Description:**

Internal ID for a script deployment record associated with a specific task.MapReduceScriptTask.

## status

**TaskStatus**

**Supported Script Types:** `Server scripts`

**Description:**

Status for a map/reduce script task. Returns a task.TaskStatus enum value.

## stage

**Task.MapReduceStage**

**Supported Script Types:** `Server scripts`

**Description:**

The current stage of a map/reduce script deployment that is being processed. See task.MapReduceStage for supported values.

---

### CsvImportTask Object Members

The following members are called on task.CsvImportTask.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>CsvImportTask.submit()</td>
<td>string</td>
<td>Server scripts</td>
<td>Directs NetSuite to place a CSV import task into the NetSuite task queue and returns a unique ID for the task.</td>
</tr>
<tr>
<td>Property</td>
<td>CsvImportTask.importFile</td>
<td>file.File</td>
<td>Server scripts</td>
<td>CSV file to import. Use a file.File object or a string that represents the CSV text to be imported.</td>
</tr>
<tr>
<td></td>
<td>CsvImportTask.mappingId</td>
<td>number</td>
<td>Server scripts</td>
<td>Script ID or internal ID of the saved import map that you created when you ran the Import Assistant.</td>
</tr>
<tr>
<td></td>
<td>CsvImportTask.queueId</td>
<td>number</td>
<td>Server scripts</td>
<td>Overrides the Queue Number property under Advanced Options on the Import Options page of the Import Assistant.</td>
</tr>
<tr>
<td></td>
<td>CsvImportTask.name</td>
<td>string</td>
<td>Server scripts</td>
<td>Name for the CSV import task.</td>
</tr>
<tr>
<td></td>
<td>CsvImportTask.linkedFiles</td>
<td>Object</td>
<td>Server scripts</td>
<td>A map of key/value pairs that sets the data to be imported in a linked file for a multi-file import job, by referencing a file in the file cabinet or the raw CSV data to import.</td>
</tr>
</tbody>
</table>

### CsvImportTaskStatus Object Members

The following members are called on task.CsvImportTaskStatus.
<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
</table>

**EntityDeduplicationTask Object Members**

The following members are called on task.EntityDeduplicationTask.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>EntityDeduplicationTask.submit</td>
<td>string</td>
<td>Server scripts</td>
<td>Directs NetSuite to place the merge duplicate records task into the NetSuite task queue and returns a unique ID for the task.</td>
</tr>
<tr>
<td>Property</td>
<td>EntityDeduplicationTask.entityType</td>
<td>task.DedupeEntityType</td>
<td>Server scripts</td>
<td>Sets the type of entity on which you want to merge duplicate records.</td>
</tr>
<tr>
<td></td>
<td>EntityDeduplicationTask.masterRecordId</td>
<td>number</td>
<td>Server scripts</td>
<td>When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.</td>
</tr>
<tr>
<td></td>
<td>EntityDeduplicationTask.masterSelectionMode</td>
<td>task.MasterSelectionMode</td>
<td>Server scripts</td>
<td>When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.</td>
</tr>
<tr>
<td></td>
<td>EntityDeduplicationTask.dedupeMode</td>
<td>task.DedupeMode</td>
<td>Server scripts</td>
<td>Sets the mode in which to merge or delete duplicate records.</td>
</tr>
<tr>
<td></td>
<td>EntityDeduplicationTask.recordIds</td>
<td>number[]</td>
<td>Server scripts</td>
<td>Number array of record internal IDs to perform the merge or delete operation on.</td>
</tr>
</tbody>
</table>

**EntityDeduplicationTaskStatus Object Members**

The following members are called on task.EntityDeduplicationTaskStatus.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>EntityDeduplicationTaskStatus.status</td>
<td>task.TaskStatus</td>
<td>Server scripts</td>
<td>Status for a merge duplicate record task.</td>
</tr>
</tbody>
</table>

**SearchTask Object Members**

The following members are called on task.SearchTask.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>SearchTask.addInboundDependency</td>
<td>void</td>
<td>Server scripts</td>
<td>Adds a scheduled script task or map/reduce script task to the search task.</td>
</tr>
</tbody>
</table>
### SearchTask Object Members

The following members are called on `Task/SearchTask`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td><code>SearchTask.fileId</code></td>
<td>number</td>
<td>Server scripts</td>
<td>ID of the CSV file to export search results into.</td>
</tr>
<tr>
<td></td>
<td><code>SearchTask.filePath</code></td>
<td>string</td>
<td>Server scripts</td>
<td>Path of the CSV file to export search results into.</td>
</tr>
<tr>
<td></td>
<td><code>SearchTask.inboundDependencies</code></td>
<td>Object</td>
<td>Server scripts</td>
<td>Object that contains key/value pairs to describe the dependent scripts added to the search task.</td>
</tr>
<tr>
<td></td>
<td><code>SearchTask.savedSearchId</code></td>
<td>number</td>
<td>Server scripts</td>
<td>ID of the saved search to be executed during the task.</td>
</tr>
</tbody>
</table>

### SearchTaskStatus Object Members

The following members are called on `Task/SearchTaskStatus`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td><code>SearchTaskStatus.fileId</code></td>
<td>number</td>
<td>Server scripts</td>
<td>ID of the CSV file into which search results are exported.</td>
</tr>
<tr>
<td></td>
<td><code>SearchTaskStatus.savedSearchId</code></td>
<td>number</td>
<td>Server scripts</td>
<td>ID of the saved search executed during the task.</td>
</tr>
<tr>
<td></td>
<td><code>SearchTaskStatus.status</code></td>
<td>task.TaskStatus</td>
<td>Server scripts</td>
<td>Status of an asynchronous search task placed in the NetSuite task queue.</td>
</tr>
<tr>
<td></td>
<td><code>SearchTaskStatus.taskId</code></td>
<td>number</td>
<td>Server scripts</td>
<td>ID of the asynchronous task.</td>
</tr>
</tbody>
</table>

### WorkflowTriggerTask Object Members

The following members are called on `Task/WorkflowTriggerTask`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>WorkflowTriggerTask.submit()</code></td>
<td>string</td>
<td>Server scripts</td>
<td>Directs NetSuite to place the asynchronous workflow initiation task into the NetSuite scheduling queue and returns a unique ID for the task.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Property</td>
<td>Workflow trigger task. record type</td>
<td>string</td>
<td>Server scripts</td>
<td>Record type of the workflow base record. For example, customer, salesorder, or lead.</td>
</tr>
<tr>
<td></td>
<td>Workflow trigger task. record ID</td>
<td>number</td>
<td>Server scripts</td>
<td>Internal ID of the workflow definition base record. For example, 55 or 124.</td>
</tr>
<tr>
<td></td>
<td>Workflow trigger task. workflow ID</td>
<td>number</td>
<td>Server scripts</td>
<td>Internal ID (as a number), or script ID (as a string), for the workflow definition.</td>
</tr>
<tr>
<td></td>
<td>Workflow trigger task. params</td>
<td>object</td>
<td>Server scripts</td>
<td>Object that contains key/value pairs to set default values on fields specific to the workflow.</td>
</tr>
</tbody>
</table>

### WorkflowTriggerTaskStatus Object Members

The following members are called on the `task.WorkflowTriggerTaskStatus` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Workflow trigger task. status</td>
<td>task.TaskStatus</td>
<td>Server scripts</td>
<td>Status for a asynchronous workflow placed in the NetSuite task queue.</td>
</tr>
</tbody>
</table>

### RecordActionTask (Beta) Object Members

**Warning:** The record action task is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature's operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

The following members are called on `task.RecordActionTaskStatus (Beta)`. 

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Record action task. submit</td>
<td>string</td>
<td>Server scripts</td>
<td>Submits a record action task for processing and returns its task ID.</td>
</tr>
<tr>
<td></td>
<td>Record action task. to string</td>
<td>string</td>
<td>Server scripts</td>
<td>Returns the object type name.</td>
</tr>
<tr>
<td></td>
<td>Record action task. to json</td>
<td>object</td>
<td>Server scripts</td>
<td>Returns an object in JSON.</td>
</tr>
<tr>
<td>Property</td>
<td>Record action task. record type</td>
<td>string</td>
<td>Server scripts</td>
<td>The record type on which the action is to be performed. For a list of record types, see <code>record.Type</code>.</td>
</tr>
<tr>
<td></td>
<td>Record action task. record ID</td>
<td>string</td>
<td>Server scripts</td>
<td>Function that takes record ID and returns the parameter object for the specified record ID.</td>
</tr>
<tr>
<td></td>
<td>Record action task. action</td>
<td>string</td>
<td>Server scripts</td>
<td>The ID of the action to be invoked.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td></td>
<td>Array of parameter objects</td>
<td>Server scripts</td>
<td>An array of parameter objects. Each object corresponds to one record ID of the record for which the action is to be executed. The object has the following form: <code>{recordId: 1, someParam: 'example1', otherParam: 'example2'}</code></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Object</td>
<td>Server scripts</td>
<td>The condition used to select record IDs of records for which the action is to be executed. Only the <code>action.ALL_QUALIFIED_INSTANCES</code> constant is currently supported.</td>
<td></td>
</tr>
</tbody>
</table>

**RecordActionTaskStatus (Beta) Object Members**

**Warning:** The record action task is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature's operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

The following members are called on `task.RecordActionTaskStatus (Beta)`.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>RecordActionTaskStatus.toObject()</code></td>
<td>Object</td>
<td>Server scripts</td>
<td>Returns an object in JSON.</td>
</tr>
<tr>
<td>Method</td>
<td><code>RecordActionTaskStatus.toJSONObject()</code></td>
<td>Object</td>
<td>Server scripts</td>
<td>Returns an object in JSON.</td>
</tr>
<tr>
<td>Property</td>
<td><code>RecordActionTaskStatus.objectType</code></td>
<td>string</td>
<td>Server scripts</td>
<td>Returns the object type name.</td>
</tr>
<tr>
<td>Property</td>
<td><code>RecordActionTaskStatus.status</code></td>
<td>string</td>
<td>Server scripts</td>
<td>Represents the record action task status. Returns a value from the <code>task.TaskStatus</code> enum.</td>
</tr>
<tr>
<td>Property</td>
<td><code>RecordActionTaskStatus.results</code></td>
<td>Object</td>
<td>Server scripts</td>
<td>The results of successfully executed record action tasks. The value of the property is the task instance ID and the corresponding action result.</td>
</tr>
<tr>
<td>Property</td>
<td><code>RecordActionTaskStatus.errors</code></td>
<td>Object</td>
<td>Server scripts</td>
<td>The error details of failed action executions. The value of the property is the record instance ID and the corresponding error details. The error details are returned in an unnamed object with two properties: code and message.</td>
</tr>
<tr>
<td>Property</td>
<td><code>RecordActionTaskStatus.completed</code></td>
<td>number</td>
<td>Server scripts</td>
<td>The number of record action tasks with a completed status.</td>
</tr>
<tr>
<td>Property</td>
<td><code>RecordActionTaskStatus.succeeded</code></td>
<td>number</td>
<td>Server scripts</td>
<td>The number of record action tasks with a succeeded status.</td>
</tr>
<tr>
<td>Property</td>
<td><code>RecordActionTaskStatus.failed</code></td>
<td>number</td>
<td>Server scripts</td>
<td>The number of record action tasks with a failed status.</td>
</tr>
</tbody>
</table>
N/task Module Script Samples

Some of the following script samples use the `require` function so that you can copy the script into the debugger and test it. However, you must use the `define` function in your entry point script (the script you attach to a script record). For more information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

Sample 1

The following script sample submits a map/reduce script deployment.

```javascript
/**
 * @NApiVersion 2.x
 */
require(["N/task", 'N/runtime', 'N/email'
function(task, runtime, email) {
  function submitMapReduceDeployment() {
    // Store the script ID of the script to submit.
    //
    // Update the following statement so it uses the script ID
    // of the map/reduce script record you want to submit.
    var mapReduceScriptId = 'customscript_test_mapreduce_script';
    log.audit('mapreduce id: ', mapReduceScriptId);
    // Create a map/reduce task.
    //
    // Update the deploymentId parameter to use the script ID of
    // the deployment record for your map/reduce script.
    var mrTask = task.create({
      taskType: task.TaskType.MAP_REDUCE,
      scriptId: mapReduceScriptId,
      deploymentId: 'customdeploy_test_mapreduce_script'
    });
    // Submit the map/reduce task.
    var mrTaskId = mrTask.submit();
    // Check the status of the task, and send an email if the
```
// task has a status of FAILED.
//
// Update the authorId value with the internal ID of the user
// who is the email sender. Update the recipientEmail value
// with the email address of the recipient.
var taskStatus = task.checkStatus(mrTaskId);
if (taskStatus.status === 'FAILED') {
  var authorId = -5;
  var recipientEmail = 'notify@myCompany.com';
  email.send({
    author: authorId,
    recipients: recipientEmail,
    subject: 'Failure executing map/reduce job!',
    body: 'Map reduce task: ' + mapReduceScriptId + ' has failed.'
  });
}

submitMapReduceDeployment();

Sample 2

The following script sample creates an asynchronous search task to execute a saved search and export the results of the search into a CSV file stored in the file cabinet. After the search task is submitted, the script retrieves the task status using the task ID.

**Note:** Some of the values in this script sample are placeholders. Before using this sample, replace all hard-coded values, such as IDs and file paths, with valid values from your NetSuite account. If you run a script with an invalid value, the system may throw an error.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/task'],
  function(task) {
    // Do one of the following:
    //
    // - Create a saved search and capture its ID. To do this, you can use
    //   the following code snippet (replacing the id, filters, and columns
    //   values as appropriate):
    //
    //   var mySearch = search.create({
    //       type: search.Type.SALES_ORDER,
    //       id: 'customsearch_my_search',
    //       filters: [...],
    //       columns: [...]
    //   });
    //   mySearch.save();
    //   var savedSearchId = mySearch.searchId;
    //
    // - Use the ID of an existing saved search. This is the approach that
    // this script sample uses. Update the following statement with the
    // internal ID of the search you want to use.
```
var savedSearchId = -10;

// Create the search task.
var myTask = task.create(
    taskType: task.TaskType.SEARCH
);  
myTask.savedSearchId = savedSearchId;

// Specify the ID of the file that search results will be exported into.
// Update the following statement so it uses the internal ID of the file you want to use.
myTask.fileId = 448;

// Submit the search task.
var myTaskId = myTask.submit();

// Retrieve the status of the search task.
var taskStatus = task.checkStatus({
    taskId: myTaskId
});

// Optionally, create new variables to represent values used previously in this script. You may want to use these variables in additional logic you add to this script.
var myFileId = taskStatus.fileId;
var mySavedSearchId = taskStatus.savedSearchId;

// Optionally, add logic that executes when the task is complete.
if (taskStatus.status === task.TaskStatus.COMPLETE) {
    // Add any code that is appropriate. For example, if this script created a saved search, you may want to delete it.
}

Sample 3

The following script sample creates a scheduled script task and a map/reduce script task. It then creates an asynchronous search task and adds the scheduled script task and the map/reduce script task to the search task as dependent scripts. These scripts are processed when the search task is complete. For more information, see the help topic SuiteCloud Processors.

Note: This script sample refers to two script parameters: custscript_ss_as_srch_res for the scheduled script, and custscript_mr_as_srch_res for the map/reduce script. These parameters are used to pass the location of the CSV file to the dependent scripts, which is shown in the second and third code snippets below. Before using this sample, create these parameters in the script record. For more information, see the help topic Creating Script Parameters.

require(['N/task'],
    function(task) {

        // Specify a file for the search results
        var asyncSearchResultFile = 'SuiteScripts/ExportFile.csv';
// Create a scheduled script task
var scheduledScript = task.create({
    taskType: task.TaskType.SCHEDULED_SCRIPT
});
scheduledScript.scriptId = 'customscript_as_ftr_ss';
scheduledScript.deploymentId = 'customdeploy_ss_dpl';
scheduledScript.params = {
    'custscript_ss_as_srch_res': asyncSearchResultFile
};

// Create a map/reduce script task
var mapReduceScript = task.create({
    taskType: task.TaskType.MAP_REDUCE
});
mapReduceScript.scriptId = 'customscript_as_ftr_mr';
mapReduceScript.deploymentId = 'customdeploy_mr_dpl';
mapReduceScript.params = {
    'custscript_mr_as_srch_res': asyncSearchResultFile
};

// Create the search task
var asyncTask = task.create({
    taskType: task.TaskType.SEARCH
});
asyncTask.savedSearchId = 'customsearch35';
asyncTask.filePath = asyncSearchResultFile;

// Add dependent scripts to the search task before it is submitted
asyncTask.addInboundDependency(scheduledScript);
asyncTask.addInboundDependency(mapReduceScript);

// Submit the search task
var asyncTaskId = asyncTask.submit();

To read the contents of the search results file within a dependent scheduled script, consider the following script sample.

 /**<
 * @NApiVersion 2.x
 * @NScriptType ScheduledScript
 */
define(['N/file', 'N/log', 'N/email', 'N/runtime'],
    function(file, log, email, runtime) {
        function execute(context) {

            // Read a CSV file and return the number of rows minus the header row.
            function numberOfRows(csvFileId) {
                var invoiceFile = file.load({
                    id: csvFileId
                });
            }
        }
    });
var iterator = invoiceFile.lines.iterator();
var noOfLines = 0;

// Skip the first row (the header row).
iterator.each(function() {
    return false;
});

// Process the rest of the rows.
iterator.each(function() {
    noOfLines++;
    return true;
});

return noOfLines;

// Send an email to the user who ran the script, and attach the CSV file with the search results.
function sendEmailWithAttachment(csvFileId) {
    var noOfRows = numberOfRows(csvFileId);
    var userId = runtime.getCurrentUser().id;
    var fileObj = file.load({
        id: csvFileId
    });

    email.send({
        author: userId,
        recipients: userId,
        subject: 'Search completed',
        body: 'CSV file attached, ' + noOfRows + ' record(s) found.',
        attachments: [fileObj]
    });
}

// Retrieve the ID of the search results file.
// Update the name parameter to use the script ID of the original search task.
var resFileId = runtime.getCurrentScript().getParameter({'custscript_ss_as_srch_res'});

if (!resFileId)
{
    log.error('Could not obtain file content from the specified ID.');
    return;
}

log.debug({
    title: 'search - numberOfRows',
    details: numberOfRows(resFileId)
});

sendEmailWithAttachment(resFileId);
To read the contents of the search results file within a dependent map/reduce script, consider the following script sample.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType MapReduceScript
 * @NModuleScope SameAccount
 */
define(['N/runtime', 'N/file', 'N/log', 'N/email'],
    // Load the search results file, count the number of letters in the file, and
    // store this count in another file.
    function(runtime, file, log, email) {

        function getInputData() {
            // Retrieve the ID of the search results file.
            // Update the completionScriptParameterName value to use the script
            // ID of the original search task.
            var completionScriptParameterName = 'custscript_mr_as_srch_res';
            var resFileId = runtime.getCurrentScript().getParameter({
                name: completionScriptParameterName
            });

            if (!resFileId) {
                log.error({
                    details: 'resFileId is not valid. Please check the script parameter stored in the
completionScriptParameterName variable in getInputData().'
                });
            }

            return {
                type: 'file',
                id: resFileId
            };
        }

        function map(context) {
            var email = context.value.split('=')[1];
            if (email) {
                var splitEmail = email.split('@');
                context.write(splitEmail[splitEmail.length-1], 1);
            }
        }

        function reduce(context) {
            context.write(context.key, context.values.length);
        }
    }
```

```javascript
return {
    execute: execute
};
```
Sample 4

The following sample submits a record action task and then checks its status.

For details about record action tasks, see task.RecordActionTask (Beta) and task.RecordActionTaskStatus (Beta).

Note: The record action task is a beta feature in 2019.1.
task.ScheduledScriptTask

**Object Description**
Encapsulates all the properties of scheduled script task in SuiteScript. Use this object to place a scheduled script deployment into the NetSuite scheduling queue.

**To use the ScheduledScriptTask Object:**

1. In the NetSuite UI, create the script record and script deployment record.
2. Use `task.create(options)` to create the ScheduledScriptTask object.
3. Use the ScheduledScriptTask object properties to set the script and deployment properties.
4. Use `ScheduledScriptTask.submit()` to deploy the scheduled script to the NetSuite scheduling queue.
5. Use the properties for the `task.ScheduledScriptTaskStatus` object to get the status of the scheduled script.

For a complete list of this object's methods and properties, see [ScheduledScriptTask Object Members](#).

For more information about scheduled scripts in NetSuite, see the help topic [SuiteScript 2.0 Scheduled Script Type](#).

**Supported Script Types**
Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/task Module

**Since**
2015.2

**Syntax**

```javascript
//Add additional code
...
var scriptTask = task.create({taskType: task.TaskType.SCHEDULED_SCRIPT});
scriptTask.scriptId = 1234;
scriptTask.deploymentId = 'customdeploy1';
scriptTask.params = {searchId: 'custsearch_456'};
var scriptTaskId = scriptTask.submit();
...`
ScheduledScriptTask.submit()

| Method Description | Directs NetSuite to place a scheduled script deployment into the NetSuite scheduling queue and returns a unique ID for the task.  
Additionally, note the following:  
- The scheduled script must have a status of Not Scheduled on the Script Deployment page. If the script status is set to Testing on the Script Deployment page, this method will not place the script into the scheduling queue.  
- If the deployment status on the Script Deployment page is set to Scheduled, the script will be placed into the queue according to the time(s) specified on the Script Deployment page.  
- Only administrators can run scheduled scripts. If a user event script calls ScheduledScriptTask.submit(), the user event script has to be deployed with admin permissions.  
- A scheduled script can be submitted for processing only if there is no unfinished scheduled script task for the same script ID and script deployment ID. Therefore, if a scheduled script resubmits itself, the actual resubmit does not occur until the current execution completes. This delay is necessary to avoid the existence of two unfinished tasks for the same deployment of the same script. For this reason, if a scheduled script uses the submit() method to resubmit itself, then at runtime, no task ID is returned when the scheduled script is submitted. |

| Returns | The task ID as a string, except as noted above. |

| Supported Script Types | Server scripts  
For more information, see the help topic SuiteScript 2.0 Script Types. |

| Governance | 20 units |
| Module | N/task Module |
| Since | 2015.2 |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_TO_SUBMIT_JOB_REQUEST_1</td>
<td>Failed to submit job request: (reason)</td>
<td>Task cannot be submitted.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var scheduledScriptTaskId = scriptTask.submit();
...
//Add additional code
```

ScheduledScriptTask.scriptId

| Property Description | Internal ID (as a number), or script ID (as a string), for the script record associated with a task.ScheduledScriptTask object. |
### ScheduledScriptTask.deploymentId

**Property Description:** Internal ID (as a number), or script ID (as a string), for the script deployment record associated with a task.ScheduledScriptTask Object.

**Type:** number | string  

**Supported Script Types:** Server scripts  
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module:** N/task Module  
**Since:** 2015.2

**Syntax**

```javascript
//Add additional code
...
var scheduledScriptId = 34;
...
//Add additional code
```

### ScheduledScriptTask.params

**Property Description:** Object with key/value pairs that override static script parameter field values on the script deployment. Use these parameters for the task.ScheduledScriptTask object to programmatically pass values to the script deployment.

---

**Type**  
number | string  

**Governance**  
20 units

**Supported Script Types**  
Server scripts  
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  
N/task Module  
**Since**  
2015.2

**Syntax**

```javascript
//Add additional code
...
scheduledTask.deploymentId = 1;
...
//Add additional code
```
### N/task Module

**Type** | object  
---|---
**Supported Script Types** | Server scripts  
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module** | N/task Module  
**Since** | 2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
scriptTask.params = {searchId: 'custsearch_456'};
...
//Add additional code
```

### task.ScheduledScriptTaskStatus

**Object Description**
Encapsulates the properties and status of a scheduled script placed into the NetSuite scheduling queue.

Use `task.checkStatus(options)` with the unique ID for the scheduled script task to get the `ScheduledScriptTaskStatus` object.

For a complete list of this object's properties, see [ScheduledScriptTaskStatus Object Members](#).

**Supported Script Types** | Server scripts  
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module** | N/task Module  
**Since** | 2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var res = task.checkStatus(scriptTaskId);
log.debug('Initial status: ' + res.status);
...
//Add additional code
```
ScheduledScriptTaskStatus.scriptId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Internal ID for a script record associated with a specific task.ScheduledScriptTask Object. Use this ID to get more details about the script record for the scheduled task.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>read-only number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Module: N/task Module

Since: 2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

```plaintext
//Add additional code
...
log.audit('Initial status: ' + status.scriptId);
...
//Add additional code
```

ScheduledScriptTaskStatus.deploymentId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Internal ID for a script deployment record associated with a specific task.ScheduledScriptTask Object. Use this ID to get more details about the script deployment record for the scheduled task.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Module: N/task Module

Since: 2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
<td></td>
</tr>
</tbody>
</table>
## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
log.audit({
  details: 'Deployment ID: ' + status.scriptId
});
...
//Add additional code
```

### ScheduledScriptTaskStatus.status

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Status for a scheduled script task. Returns a task.TaskStatus enum value.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>task.TaskStatus</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/task Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
log.audit({
  details: 'Status: ' + summary.status
});
...
//Add additional code
```

### task.MapReduceScriptTask

**Object Description**

Encapsulates the properties of a map/reduce script deployment. You can use this object to programmatically submit a script deployment for processing.
To use the `MapReduceScriptTask` object:

- In the NetSuite UI, create the script record and script deployment records.
- Use `task.create(options)` to create the `MapReduceScriptTask` object.
- Use the `MapReduceScriptTask` object properties to set the script and deployment properties.
- Use `MapReduceScriptTask.submit()` to submit the deployment for processing.
- Use the properties for the `task.MapReduceScriptTaskStatus` object to get the status of the map/reduce script.

For a complete list of this object's methods and properties, see `MapReduceScriptTask Object Members`.

For general information about map/reduce scripts, see the help topic `Map/Reduce Key Concepts`.

**Supported Script Types**

**Server scripts**

For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Module**

N/task Module

**Since**

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code...
var mrTask = task.create({taskType: task.TaskType.MAP_REDUCE});
mrTask.scriptId = mapReduceScriptId;
mrTask.deploymentId = 1;
var mrTaskId = mrTask.submit();
...//Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic `SuiteScript 2.0 Map/Reduce Script Type`.

### MapReduceScriptTask.submit()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Submits a map/reduce script deployment for processing. For more information, see <code>task.MapReduceScriptTask</code>.</th>
</tr>
</thead>
</table>

Additionally, note that a map/reduce script can be submitted for processing only if there is no unfinished map/reduce script task for the same script ID and script deployment ID. For this reason, if a map/reduce script resubmits itself, the actual resubmit does not occur until the current execution completes. This delay is necessary to avoid the existence of two unfinished tasks for the same deployment of the same script. Therefore, if a map/reduce script uses the `submit()` method to resubmit itself, then at runtime, no task ID is returned when the map/reduce script is submitted.

For general information about the execution of map/reduce scripts, see the help topic `SuiteScript 2.0 Map/Reduce Script Submission`. 
Returns | string
---|---
**Supported Script Types** | Server scripts
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

Governance | 20 units
Module | N/task Module
Since | 2015.2

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_TO_SUBMIT_JOB_REQUEST_1</td>
<td>Failed to submit job request: {reason}</td>
<td>Task cannot be submitted.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var mrTaskId = mrTask.submit();
...
//Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

**MapReduceScriptTask.scriptId**

**Property Description** | Internal ID (as a number), or script ID (as a string), for the map/reduce script record.
---|---
**Type** | number | string
**Supported Script Types** | Server scripts
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

Module | N/task Module
Since | 2015.2

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
```
var mapReduceScriptId = 34;

//Add additional code

Note: For general information about map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Type.

MapReduceScriptTask.deploymentId

Property Description: Internal ID (as a number) or script ID (as a string), for the script deployment record for a map/reduce script.

Type: number | string

Supported Script Types: Server scripts

Module: N/task Module

Since: 2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

//Add additional code
...

mrTask.deploymentId = 1;
...

//Add additional code

Note: For general information about map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Type.

MapReduceScriptTask.params

Property Description: Object that represents key/value pairs that override static script parameter field values on the script deployment record. Use these parameters on a task.MapReduceScriptTask object to programmatically pass values to the script deployment. For more information about script parameters, see the help topic Creating Script Parameters Overview.

Type: object

Supported Script Types: Server scripts

Module: N/task Module

Since: 2015.2
task.MapReduceScriptTaskStatus

Object Description
Encapsulates the status of a map/reduce script deployment that was submitted for processing.

Use `task.checkStatus(options)` with the unique ID for the map/reduce script task to get the `task.MapReduceScriptTaskStatus` object.

For a complete list of this object's methods and properties, see `task.MapReduceScriptTaskStatus` Object Members.

For general information about the execution of map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Submission.

Supported Script Types
Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/task Module

Since
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...

mrTask.params = {doSomething: true};
...
//Add additional code

var summary = task.checkStatus(scriptTaskId);
if (summary.stage === task.MapReduceStage.SUMMARIZE)
  log.audit('Almost done...');
...
//Add additional code
```
MapReduceScriptTaskStatus.getPercentageCompleted()

Method Description
Returns the current percentage complete for the current stage of a task.MapReduceScriptTask.
Use the MapReduceScriptTaskStatus.stage property to get the current stage.
For general information about map/reduce stages, see the help topics Map/Reduce Key Concepts and SuiteScript 2.0 Map/Reduce Script Stages.

Note: The input and summarize stages are either 0% or 100% complete at any time.

Returns
number

Supported Script Types
Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 units

Module
N/task Module

Since
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var completion = taskStatus.getPercentageCompleted();
log.audit('Percentage Completed: ' + completion);
...
//Add additional code
```

Note: For general information about map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Type.

MapReduceScriptTaskStatus.getPendingMapCount()

Method Description
Returns the total number of records or rows not yet processed by the map stage of a task.MapReduceScriptTask.
Use the MapReduceScriptTaskStatus.stage property to get the current stage.
For general information about map/reduce stages, see the help topics Map/Reduce Key Concepts and SuiteScript 2.0 Map/Reduce Script Stages.

Returns
number
Supported Script Types | Server scripts  
| For more information, see the help topic SuiteScript 2.0 Script Types.

Governance | 10 units

Module | N/task Module

Since | 2015.2

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = taskStatus.getPendingMapCount();
log.audit('Pending Map Count: ' + summary);
...
//Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Type.

MapReduceScriptTaskStatus.getTotalMapCount()

**Method Description**

Returns the total number of records or rows passed as input to the map stage of a task.MapReduceScriptTask.

Use the MapReduceScriptTaskStatus.stage property to get the current stage.

For general information about map/reduce stages, see the help topics Map/Reduce Key Concepts and SuiteScript 2.0 Map/Reduce Script Stages.

**Returns** | number

**Supported Script Types** | Server scripts  
| For more information, see the help topic SuiteScript 2.0 Script Types.

Governance | 10 units

Module | N/task Module

Since | 2015.2

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = taskStatus.getTotalMapCount();
log.audit('Total Map Count: ' + summary);
```
### MapReduceScriptTaskStatus.getPendingMapSize()

**Method Description**

Returns the total number of bytes not yet processed by the map stage, as a component of total size, of a `task.MapReduceScriptTask`.

Use the `MapReduceScriptTaskStatus.stage` property to get the current stage.

For general information about map/reduce stages, see the help topics [Map/Reduce Key Concepts](#) and [SuiteScript 2.0 Map/Reduce Script Stages](#).

**Returns**

`number`

**Supported Script Types**

Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

25 units

**Module**

N/task Module

**Since**

2015.2

**Syntax**

```javascript
//Add additional code
...
var summary = taskStatus.getPendingMapSize();
log.audit('Pending Map Size: ' + summary);
...
//Add additional code
```

### MapReduceScriptTaskStatus.getPendingReduceCount()

**Method Description**

Returns the total number of records or rows not yet processed by the reduce stage of a `task.MapReduceScriptTask`.

Use the `MapReduceScriptTaskStatus.stage` property to get the current stage.

For general information about the reduce stage and other map/reduce stages, see the help topics [Map/Reduce Key Concepts](#) and [SuiteScript 2.0 Map/Reduce Script Stages](#).

**Returns**

`number`

**Supported Script Types**

Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Governance</th>
<th>10 units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = taskStatus.getPendingReduceCount();
log.audit({
  details: 'Pending Reduce Count: ' + summary
});
...
//Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Type.

MapReduceScriptTaskStatus.getTotalReduceCount()

**Method Description**

Returns the total number of record or row inputs to the reduce stage of a MapReduceScriptTask.

Use the MapReduceScriptTaskStatus.stage property to get the current stage.

For general information about map/reduce stages, see the help topics Map/Reduce Key Concepts and SuiteScript 2.0 Map/Reduce Script Stages.

**Returns**

number

**Supported Script Types**

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Governance</th>
<th>10 units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = taskStatus.getTotalReduceCount();
log.audit({
  details: 'Reduce Count: ' + summary
});
```
```javascript
//Add additional code
...

// Add additional code
var summary = taskStatus.getPendingReduceSize();
log.audit({
    details: 'Pending Reduce Size: ' + summary
});
...
//Add additional code
```

**MapReduceScriptTaskStatus.getPendingReduceSize()**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the total number of bytes not yet processed by the reduce stage, as a</td>
<td>number</td>
<td>Server scripts</td>
<td>25 units</td>
<td>N/task Module</td>
<td>2015.2</td>
</tr>
<tr>
<td>component of total size, of a task.MapReduceScriptTask.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Use the MapReduceScriptTaskStatus.stage property to get the current stage.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**MapReduceScriptTaskStatus.getPendingOutputCount()**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns the total number of records or rows not yet processed by a task.MapReduceScriptTask.</td>
<td>number</td>
<td>Server scripts</td>
<td>10 units</td>
<td></td>
<td></td>
</tr>
<tr>
<td>For more information, see the help topic <strong>SuiteScript 2.0 Script Types.</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Note:** For general information about map/reduce scripts, see the help topic **SuiteScript 2.0 Map/Reduce Script Type.**
### MapReduceScriptTaskStatus.getPendingOutputSize()

**Method Description**  
Returns the total size in bytes of all key/value pairs written as output, as a component of total size, by a `task.MapReduceScriptTask`.

**Returns**  
`number`

**Supported Script Types**  
Server scripts  
For more information, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

**Governance**  
25 units

**Module**  
N/task Module

**Since**  
2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getPendingOutputSize();
log.audit({
    title: 'Size',
    details: total
});
...
//Add additional code
```
MapReduceScriptTaskStatus.getTotalOutputCount()

**Method Description**
Returns the total number of key/value pairs passed as inputs to the **SUMMARIZE** phase of a `task.MapReduceScriptTask`. Use the `MapReduceScriptTaskStatus.stage` property to get the current stage.

**Returns**
number

**Supported Script Types**
Server scripts
For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Governance**
10 units

**Module**
N/task Module

**Since**
2015.2

**Syntax**

```
//Add additional code

var summary = task.checkStatus(scriptTaskId);
var total = summary.getTotalOutputCount();
log.audit({
    title: 'Total Entries Passed to Output',
    details: total
});

//Add additional code
```

Note: For general information about map/reduce scripts, see the help topic `SuiteScript 2.0 Map/Reduce Script Type`.

MapReduceScriptTaskStatus.getCurrentTotalSize()

**Method Description**
Returns the total size in bytes of all stored work in progress by a `task.MapReduceScriptTask`.

**Returns**
number

**Supported Script Types**
Server scripts
For more information, see the help topic `SuiteScript 2.0 Script Types`.

**Governance**
25 units
Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
var total = summary.getCurrentTotalSize()
log.audit({
    title: 'Size of Remaining Data to Process',
    details: total
});
...
//Add additional code
```

Note: For general information about map/reduce scripts, see the help topic SuiteScript 2.0 Map/Reduce Script Type.

MapReduceScriptTaskStatus.scriptId

Property Description: Internal ID for a map/reduce script record associated with a specific task.MapReduceScriptTask.

Type: read-only number

Supported Script Types: Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module: N/task Module

Since: 2015.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Script ID',
```
```javascript
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Deployment ID',
    details: summary.deploymentId
});
```

#### Error Code

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
<td></td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```
var summary = task.checkStatus(scriptTaskId);
log.audit({
    title: 'Deployment ID',
    details: summary.deploymentId
});
```

Note: For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).
For general details about the execution of map/reduce scripts, see the help topic "SuiteScript 2.0 Map/Reduce Script Submission."

<table>
<thead>
<tr>
<th>Type</th>
<th>task.TaskStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic &quot;SuiteScript 2.0 Script Types.&quot;</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td></td>
<td>Setting the property is attempted</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
logaudit{
    title: 'Status',
    details: summary.status
});
...
//Add additional code
```

Note: For general information about map/reduce scripts, see the help topic "SuiteScript 2.0 Map/Reduce Script Type."

### MapReduceScriptTaskStatus.stage

**Property Description**

Current stage of processing for a map/reduce script deployment instance. See `task.MapReduceStage` for supported values.

For general information about map/reduce stages, see the help topic "Map/Reduce Key Concepts." For information about the execution of map/reduce scripts, see the help topic "SuiteScript 2.0 Map/Reduce Script Submission."

<table>
<thead>
<tr>
<th>Type</th>
<th>task.MapReduceStage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic &quot;SuiteScript 2.0 Script Types.&quot;</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
if (summary.stage === task.MapReduceStage.SUMMARIZE)
  log.audit({
    details: 'Almost done...
  });
...
//Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

### task.CsvImportTask

**Object Description**

Encapsulates the properties of a CSV import task. Use the methods and properties for this object to submit a CSV import task into the task queue and asynchronously import record data into NetSuite.

Use the `CsvImportTask` Object to perform the following types of tasks:

- Automate standard record data import for SuiteApp installations, demo environments, and testing environments.
- Import data on a schedule using a scheduled script.
- Build integrated CSV imports with RESTlets.

Use the following process to import CSV data with `CsvImportTask`:

- In the NetSuite UI, run the Import Assistant to set up the CSV mapping and import options. You must run the Import Assistant to set up the necessary mapping for the CSV import. You can use a sample file or files to set up the mapping. Note the following information:
  - Script ID for import map.
  - Any required linked files.
- For more information, see the help topic [Importing CSV Files with the Import Assistant](#).

- Use `task.create(options)` to create the `CsvImportTask` object.
- Use the `CsvImportTask` object properties to set the script and deployment properties.
- Use `CsvImportTask.submit()` to submit the import task to the NetSuite task queue.
- Use the properties for the `task.CsvImportTaskStatus` object to get the status of the import process.

Use the following guidelines with the `CsvImportTask` Object:

- CSV imports performed within scripts are subject to the existing application limit of 25,000 records.
- You cannot import data that is imported by (2-step) assistants in the UI, because these import types do not support saved import maps. This limitation applies to budget, single journal entry, single inventory worksheet, project tasks, and website redirects imports.
This object has access only to the field mappings of a saved import map; it does not have access to advanced import options defined in the Import Assistant, such as multi-threading and multiple queues.

Even if you set options to use multiple threads or queues for an import job and then save the import map, these settings are not available to `CsvImportTask`. When this object submits a CSV import job based on the saved import map, a single thread and single queue are used.

For a complete list of this object's methods and properties, see `CsvImportTask` Object Members.

**Supported Script Types**

- Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/task Module

**Since**

2015.2

---

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var scriptTask = task.create({taskType: task.TaskType.CSV_IMPORT});
scriptTask.mappingId = 51;
var f = file.load('SuiteScripts/custjoblist.csv');
scriptTask.importFile = f;
scriptTask.linkedFiles = {'addressbook': 'street,city\nval1,val2', 'purchases': file.load('SuiteScripts/other.csv')};
var csvImportTaskId = scriptTask.submit();
...
//Add additional code
```

### `CsvImportTask.submit()`

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Directs NetSuite to place a CSV import task into the NetSuite task queue and returns a unique ID for the task. Use <code>CsvImportTaskStatus.status</code> to view the status of a submitted task. This method throws errors resulting from inline validation of CSV file data before the import of data begins (the same validation that is performed between the mapping step and the save step in the Import Assistant). Any errors that occur during the import job are recorded in the CSV response file, as they are for imports initiated through the Import Assistant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance | 100 units |
| Module | N/task Module |
| Since | 2015.2 |
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_TO_SUBMIT_JOB_REQUEST_1</td>
<td>Failed to submit job request: {reason}</td>
<td>Task cannot be submitted.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var csvImportTaskId = csvTask.submit();
...
//Add additional code
```

### CsvImportTask.importFile

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSV file to import. Use a file.File object or a string that represents the CSV text to be imported.</td>
<td></td>
</tr>
</tbody>
</table>

**Type**

- file.File
- string

**Supported Script Types**

- Server scripts
  - For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

- N/task Module

**Since**

- 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var f = file.load('SuiteScripts/custjoblist.csv');
scriptTask.importFile = f;
...
//Add additional code
```

### CsvImportTask.mappingId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Script ID or internal ID of the saved import map that you created when you ran the Import Assistant. See task.CsvImportTask.</td>
<td></td>
</tr>
</tbody>
</table>

**Type**

- number
- string

**Supported Script Types**

- Server scripts
  - For more information, see the help topic SuiteScript 2.0 Script Types.
## Important

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
scriptTask.mappingId = 51;
...
//Add additional code
```

### CsvImportTask.queueId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overrides the Queue Number property under Advanced Options on the Import Options page of the Import Assistant. Use this property to programmatically select an import queue and improve performance during the import.</td>
</tr>
</tbody>
</table>

**Note:** This property is only available if you have a SuiteCloud Plus license. For more information about using multiple queues when importing CSV files, see the help topics Queue Number and Use Multiple Threads and Multiple Queues to Run CSV Import Jobs.

<table>
<thead>
<tr>
<th>Type</th>
<th>number</th>
</tr>
</thead>
</table>

### Supports Script Types

Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

### Since

2015.2

---

## Important

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
scriptTask.queueId = 2;
...
//Add additional code
```

### CsvImportTask.name

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Name for the CSV import task. You can optionally set a different name for a scripted import task. In the UI, this name appears on the CSV Import Job Status page.</td>
</tr>
</tbody>
</table>

---

**Module** | N/task Module  
**Since** | 2015.2

---

SuiteScript 2.0 API Reference
## CsvImportTask.linkedFiles

**Property Description**

A map of key/value pairs that sets the data to be imported in a linked file for a multi-file import job, by referencing a file in the file cabinet or the raw CSV data to import.

The key is the internal ID of the record sublist for which data is being imported and the value is either a file.File object or the raw CSV data to import.

You can assign multiple types of values to the linkedFiles property.

**Type**

Object

**Supported Script Types**

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
csvTask.name = 'Import Entities'
...
//Add additional code
```

## task.CsvImportTaskStatus

**Object Description**

Encapsulates the status of a CSV import task placed into the NetSuite scheduling queue.
Use `task.checkStatus(options)` with the unique ID for the CSV import task to get the `CsvImportTaskStatus` object.

For a complete list of this object's properties, see `CsvImportTaskStatus Object Members`.

**Supported Script Types**

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/task Module

**Since**

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var csvTaskStatus = task.checkStatus({
    taskId: csvTaskId
});
if (csvTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

### CsvImportTaskStatus.status

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Status for a CSV import task. Returns a <code>task.TaskStatus</code> enum value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td><code>task.TaskStatus</code></td>
</tr>
</tbody>
</table>

**Supported Script Types**

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/task Module

**Since**

2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
```
### task.EntityDeduplicationTask

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates all the properties of a merge duplicate records task request. Use the methods and properties of this object to submit a merge duplicate record job task into the NetSuite task queue.</th>
</tr>
</thead>
</table>

When you submit a merge duplicate record task to NetSuite, SuiteScript enables you to use all of the same functionality available through the UI. Use SuiteScript to use the predefined duplicate detection rules, or you can define your own. After the records are merged or deleted, in the UI, the records no longer appear as duplicates at Lists > Mass Update > Entity Duplicate Resolution Lists > Mass Update > Entity Duplicate Resolution.

For more information about merging duplicate records in NetSuite, see the help topic Merging or Deleting Duplicate Records.

To use the `EntityDeduplicationTask` Object:

- Use `task.create(options)` to create the `EntityDeduplicationTask` object.
- Use `EntityDeduplicationTask.entityType` to select the entity type on which you want to merge duplicate records.
- Use `EntityDeduplicationTask.dedupeMode` to select the action to take for the duplicate records.
- Use a `EntityDeduplicationTask.masterSelectionMode` enum value to identify which record to use as the master record in the merge.
- If you use `MasterSelectionMode.SELECT_BY_ID` for the master selection mode, set the ID of the master record with `EntityDeduplicationTask.masterRecordId`.
- Identify the duplicate records. Use the `search.duplicates(options)` method in the `N/search Module` to find the duplicate records.
- Use `EntityDeduplicationTask.submit()` to submit the merge duplicate record task to the NetSuite task queue.
- Use the properties for the `task.EntityDeduplicationTaskStatus` object to get the status of the merge duplicate record task.

Use the following guidelines with the `EntityDeduplicationTask` Object:

- You can only submit 200 records in a single merge duplicate records task.
- The merge duplicate functionality on non-entity records is not supported in SuiteScript.
- You must have full access to the Duplicate Record Management permission to merge duplicates.

For a complete list of this object's methods and properties, see `EntityDeduplicationTask Object Members`.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server scripts</th>
</tr>
</thead>
</table>

For more information, see the help topic `SuiteScript 2.0 Script Types`.  

```javascript
...  
var summary = task.checkStatus({
  taskId: scriptTaskId
});
log.audit({
  title: 'Status',
  details: summary.status
});
...
// Add additional code
```
### Module

<table>
<thead>
<tr>
<th>Module</th>
<th>N/task Module</th>
</tr>
</thead>
</table>

| Since    | 2015.2        |

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var dedupeTask = task.create({taskType: task.TaskType.ENTITY_DEDUPLICATION});
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
dedupeTask.recordIds = ['107', '110'];
var dedupeTaskId = dedupeTask.submit();
...
//Add additional code
```

### EntityDeduplicationTask.submit()

#### Method Description

Directs NetSuite to place the merge duplicate records task into the NetSuite task queue and returns a unique ID for the task.

Use `EntityDeduplicationTaskStatus.status` to view the status of a submitted task.

#### Returns

task id as a string

#### Supported Script Types

Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

#### Governance

100 units

#### Module

N/task Module

#### Since

2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_TO_SUBMIT_JOB_REQUEST_1</td>
<td>Failed to submit job request: {reason}</td>
<td>Task cannot be submitted.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var dedupeTaskId = dedupeTask.submit();
```
EntityDeduplicationTask.entityType

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Sets the type of entity on which you want to merge duplicate records. Use a task.DedupeEntityType enum value to set the value.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>If you set entityType to CUSTOMER, the system will automatically include prospects and leads in the task request.</td>
</tr>
</tbody>
</table>

**Type**

- task.DedupeEntityType

**Supported Script Types**

- Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

- N/task Module

**Since**

- 2015.2

**Syntax**

```javascript
//Add additional code
...
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
...
//Add additional code
```

EntityDeduplicationTask.masterRecordId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record. Use this property to set the ID of the master record that you want to use as the master record in the merge.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Important:</strong></td>
<td>You must also select SELECT_BY_ID for the EntityDeduplicationTask.masterSelectionMode property, or NetSuite ignores this setting.</td>
</tr>
</tbody>
</table>

**Type**

- number

**Supported Script Types**

- Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

- N/task Module

**Since**

- 2015.2
### EntityDeduplicationTask.masterSelectionMode

**Property Description**
When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record.

Set this property to determine which of the duplicate records to keep or select the master record to use by ID.

Use `EntityDeduplicationTask.masterSelectionMode` to set the value.

**Type**
task.MasterSelectionMode

**Supported Script Types**
Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Since**
2015.2

### EntityDeduplicationTask.dedupeMode

**Property Description**
Sets the mode in which to merge or delete duplicate records.

Use a `EntityDeduplicationTask.dedupeMode` enum value to set the value.

**Type**
task.DedupeMode

**Supported Script Types**
Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.
## Module

### N/task Module

### Since

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...
//Add additional code
```

### EntityDeduplicationTask.recordIds

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Number array of record internal IDs to perform the merge or delete operation on. You can use the <code>search.duplicates(options)</code> method to identify duplicate records or create an array with record internal IDs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td><code>number[]</code></td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts  For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
</tbody>
</table>

### Module

### N/task Module

### Since

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
dedupeTask.recordIds = ['107', '110'];
...
//Add additional code
```

### task.EntityDeduplicationTaskStatus

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the status of a merge duplicate record task placed into the NetSuite task queue by <code>EntityDeduplicationTask.submit()</code>. Use <code>task.checkStatus(options)</code> with the unique ID for the merge duplicate records task to get this Object. For a complete list of this object's properties, see <a href="#">EntityDeduplicationTaskStatus Object Members</a>.</th>
</tr>
</thead>
</table>
**Supported Script Types**

<table>
<thead>
<tr>
<th>Module</th>
<th>N/task Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var dedupeTaskStatus = task.checkStatus({
  taskId: taskId
});
if (dedupeTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

### EntityDeduplicationTaskStatus.status

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Status for a merge duplicate record task. Returns a <code>task.TaskStatus</code> enum value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td><code>task.TaskStatus</code></td>
</tr>
</tbody>
</table>
| Supported Script Types | Server scripts
  For more information, see the help topic SuiteScript 2.0 Script Types. |

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
<td></td>
</tr>
</tbody>
</table>

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = task.checkStatus({
  taskId: scriptTaskId
});
```
log.audit({
  title: 'Status',
  details: summary.status
});
...
//Add additional code

task.SearchTask

Object Description
Encapsulates the properties of a search task. Use the methods and properties for this object to submit a search task into the task queue, execute it asynchronously, and persist search results. Similar to SuiteAnalytics persisted search functionality, this capability is useful for searches across high volumes of data.

You can create a task.SearchTask object using task.create(options).

Use the task.SearchTask object to do the following:
- Set the search ID using the SearchTask.savedSearchId property.
- Set the file ID or file path of a CSV file in the File Cabinet. Search results are exported to this file. Use the SearchTask.filePath property or the SearchTask.filePath property. Exactly one of these properties must be set. If both are set, an error occurs.
- Add dependent scripts to the search task using SearchTask.addInboundDependency(). Dependent scripts are processed automatically when the search task is complete.
- Submit the search task to the NetSuite task queue using SearchTask.submit().
- Get the status of a search task using the properties of the task.SearchTaskStatus object.

Note: There a limit to the number of asynchronous searches running at the same time. The limit is set to be the same as the limit for CSV import. The file size limit is based on File Cabinet limits.

| Supported Script Types | Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Module</th>
<th>N/task Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2017.1</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var searchTask = task.create({
  taskType: task.TaskType.SEARCH
});
searchTask.savedSearchId = 51;

var path = 'ExportFolder/export.csv';
searchTask.filePath = path;

var searchTaskId = searchTask.submit();
...
//Add additional code
```
SearchTask.addInboundDependency()

## Method Description

Adds a scheduled script task (task.ScheduledScriptTask) or map-reduce script task (task.MapReduceScriptTask) to the search task as a dependent script. Dependent scripts are processed automatically when the search task is complete. For more information, see the help topic SuiteCloud Processors.

**Note:** You can add only scheduled scripts or map/reduce scripts as dependent scripts to asynchronous search tasks. Other script types are not supported.

When you use this method to add a dependent script, the script is considered an inbound dependency of the search task. The added script depends on the search task. For example, if you add a scheduled script task as a dependent script, the scheduled script depends on the search task. Because `addInboundDependency()` is called on the search task, any dependent scripts that you add are considered inbound dependencies.

### Returns

**void**

### Supported Script Types

- **Server scripts**
  
  For more information, see the help topic SuiteScript 2.0 Script Types.

### Governance

None

### Module

N/task Module

### Since

2018.2

## Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>dependentScript</td>
<td>task.ScheduledScriptTask</td>
<td>Required</td>
<td>The script to add as a dependent script to the search task. Use <code>task.create(options)</code> and the <code>TaskType</code> enum to create a script task with a type of <code>SCHEDULED_SCRIPT</code> or <code>MAP_REDUCE</code>. This script task is a <code>task.ScheduledScriptTask</code> object or a <code>task.MapReduceScriptTask</code>, and you can add this script task as a dependent script to the search task. The dependent script is processed when the search task is complete. You can add only one dependent script per call to <code>SearchTask.addInboundDependency()</code></td>
<td>2018.2</td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
var scheduledScript = task.create({
```
SearchTask.submit()

Method Description
Directs NetSuite to initiate the asynchronous search task and return a unique ID for the task. When the submission is successful, this method adds the internal IDs of any dependent scripts (added using SearchTask.addInboundDependency()) to the SearchTask.inboundDependencies property.

Use task.SearchTaskStatus to view the status of a submitted task.

Returns
The task ID as a string

Supported Script Types
Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
100 units

Module
N/task Module

Since
2017.1

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_TO_SUBMIT_JOB_REQUEST_1</td>
<td>Task cannot be submitted.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A required parameter is missing.</td>
</tr>
<tr>
<td>YOU_DO_NOT_HAVE_ACCESS_TO_THE_MEDIA_ITEM</td>
<td>You do not have permission to access the file.</td>
</tr>
<tr>
<td>THAT_RECORD_DOES_NOT_EXIST</td>
<td>The file Object references a file that doesn't exist.</td>
</tr>
<tr>
<td>MUST_IDENTIFY_A_FILE</td>
<td>The path specifies a folder and not a file.</td>
</tr>
<tr>
<td>CANNOT_RESUBMIT_SUBMITTED_ASYNC_SEARCH_TASK</td>
<td>The search task was already submitted and completed successfully.</td>
</tr>
<tr>
<td>ASYNC_SEARCH_DEPENDENCY_MR_ALREADY_SUBMITTED</td>
<td>A dependent map/reduce script is already submitted and is not complete.</td>
</tr>
</tbody>
</table>
## Error Code

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASYNC_SEARCH_DEPENDENCY_MR_INCORRECT_STATUS</td>
<td>The status of the deployment record for the specified dependent map/reduce script has a value other than “Not Scheduled”.</td>
</tr>
<tr>
<td>ASYNC_SEARCH_DEPENDENCY_SS_ALREADY_SUBMITTED</td>
<td>A dependent scheduled script is already submitted and is not complete.</td>
</tr>
<tr>
<td>ASYNC_SEARCH_DEPENDENCY_SS_INCORRECT_STATUS</td>
<td>The status of the deployment record for the specified dependent scheduled script has a value other than “Not Scheduled”.</td>
</tr>
<tr>
<td>ASYNC_SEARCH_DEPLOYMENT_FOR_DEPENDENCY</td>
<td>A deployment record for the specified dependent script is not available for one of the following reasons:</td>
</tr>
<tr>
<td>■ A deployment record was not specified when the dependent script was created, and automatic lookup for an available deployment record failed.</td>
<td></td>
</tr>
<tr>
<td>■ The deployment record specified when the dependent script was created is not found.</td>
<td></td>
</tr>
<tr>
<td>ASYNC_SEARCH_MULTIPLE_DEPENDENCIES</td>
<td>The same dependent script is passed to this method more than once.</td>
</tr>
<tr>
<td>ASYNC_SEARCH_SCRIPT_ID_NOT_FOUND</td>
<td>The specified dependent script is not found.</td>
</tr>
<tr>
<td>ASYNC_SEARCH_SEARCH_ID_NOT_FOUND</td>
<td>The search task with the specified search ID is not found.</td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var searchTriggerTask = searchTask.submit();
...
//Add additional code
```

### SearchTask.filePath

<table>
<thead>
<tr>
<th>Property Description</th>
<th>ID of the CSV file to export search results into.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>Either this property or the <code>SearchTask.filePath</code> property must be set. If both are set, an error occurs.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>The CSV file ID as a number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
</tbody>
</table>

**Module**

N/task Module
### SearchTask.filePath

**Property Description**
Path of the CSV file to export search results into.

**Note:** Either this property or the `SearchTask.fileId` property must be set. If both are set, an error occurs.

**Type**
The CSV file path as a string

**Supported Script Types**
Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](https://oracle.com).

**Module**
N/task Module

**Since**
2017.1

---

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNSUPPORTED_COMBINATION_OF_PARAMETERS</td>
<td>Both this property and the <code>SearchTask.filePath</code> property are set at the same time.</td>
</tr>
</tbody>
</table>

---

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](https://oracle.com).

```javascript
//Add additional code
...
searchTask.fileId = 18;
...
//Add additional code
```

---

```javascript
//Add additional code
...
searchTask.filePath = 'ExportFolder/export.csv'
```
SearchTask.inboundDependencies

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Object with key/value pairs that contain information about the dependent scripts added to the search task. Use this property to verify the properties of dependent scripts after you add the scripts using SearchTask.addInboundDependency(). This property uses nested objects to store information about each dependent script. A nested object is included for each dependent script added to the search task. The nested object contains information such as the task type, script ID, and deployment ID. It also includes the index of the script (starting at 0). Dependent scripts are indexed in the order they are added to the search task. For example, consider a situation in which you add a scheduled script task and a map/reduce script task to a search task as dependent scripts. After you add the dependent scripts, but before you submit the search task using SearchTask.submit(), the value of the SearchTask.inboundDependencies property is similar to the following:</td>
</tr>
<tr>
<td></td>
<td>After you submit the search task, the internal IDs of the dependent scripts are added to the SearchTask.inboundDependencies property:</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Type**

read-only Object[]

**Supported Script Types**

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/task Module

---

...
Since 2018.2

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY_PROPERTY</td>
<td>Setting the property is attempted</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
var scheduledScript = task.create({
  taskType: task.TaskType.SCHEDULED_SCRIPT
});
// Set the properties of the scheduled script task
scheduledScript.scriptId = 'customscript_as_ftr_ss';
...

var mapReduceScript = task.create({
  taskType: task.TaskType.MAP_REDUCE
});
// Set the properties of the map/reduce script task
mapReduceScript.scriptId = 'customscript_as_ftr_mr';
...

asyncTask.addInboundDependency(scheduledScript);
asyncTask.addInboundDependency(mapReduceScript);

var asyncTaskId = asyncTask.submit();

// Iterate over the dependent scripts
var p = asyncTask.inboundDependencies;
for (var key in p) {
  log.debug(key + ' > ' + p[key]);
}
...
// Add additional code
```

**SearchTask.savedSearchId**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>ID of the saved search to be executed during the task.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The saved search ID as a number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

Module N/task Module

Since 2017.1
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
// Add additional code
...
searchTask.savedSearchId = 51;
...
// Add additional code
```

**task.SearchTaskStatus**

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the status of an asynchronous search task (task.SearchTask) placed into the NetSuite task queue. To initiate the task and retrieve the task ID, use SearchTask.submit().</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2017.1</td>
</tr>
</tbody>
</table>

**SearchTaskStatus.fileId**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>ID of CSV file into which search results are exported.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>CSV file id as a number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2017.1</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td></td>
<td>Setting the property is attempted</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var status = task.checkStatus(
    searchTaskId: 81
);
log.audit(
    title: 'File ID',
    details: status.fileId
);
...
//Add additional code
```

**SearchTaskStatus.savedSearchId**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The ID of the saved search executed during the task.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The search ID as a number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

**Module**

N/task Module

**Since**

2017.1

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td></td>
<td>Setting the property is attempted</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var status = task.checkStatus(
    searchTaskId: 81
);
log.audit(
    title: 'Saved Search ID',
    details: status.savedSearchId
);
...
//Add additional code
```
SearchTaskStatus.status

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Status for an asynchronous search placed in the NetSuite task queue by SearchTask.submit(). Returns a task.TaskStatus enum value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>task.TaskStatus</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts  For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2017.1</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit({
  title: 'Status',
  details: summary.status
});
...
//Add additional code
```

SearchTaskStatus.taskId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>ID of the task.TaskSearch Object. Use SearchTask.submit() to return this ID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts  For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Since</td>
<td>2017.1</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td></td>
<td>Setting the property is attempted</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var searchTaskId = searchTask.submit();
...
//Add additional code
```

task.WorkflowTriggerTask

Object Description

Encapsulates all the properties required to asynchronously initiate a workflow. Use the WorkflowTriggerTask Object to create a task that initiates an instance of the specified workflow.

The task is placed in the scheduling queue, and the workflow instance is initiated after the task reaches the top of the queue.

To use the WorkflowTriggerTask Object:

- Use `task.create(options)` to create the WorkflowTriggerTask Object.
- Use `WorkflowTriggerTask.recordType` to set the record type of the workflow base record.
- Use `WorkflowTriggerTask.recordId` to set the internal ID of the base record for the workflow.
- Use `WorkflowTriggerTask.workflowId` to set the internal ID of the workflow that you want to run on the record specified by the recordId.
- Optionally, use `WorkflowTriggerTask.params` to specify default values for workflow fields.
- Use `WorkflowTriggerTask.submit()` to submit the asynchronous workflow initiation task to the NetSuite task queue.
- Use the properties for the WorkflowTriggerTaskStatus.status object to get the status of the workflow execution.

Use the following guidelines with the WorkflowTriggerTask Object:

- `WorkflowTriggerTask.submit()` does not successfully place a workflow task in the scheduling queue if an identical instance of that workflow, with the same recordType, recordId, and workflowId, is currently executing or already in the scheduling queue.

For a complete list of this object's methods and properties, see WorkflowTriggerTask Object Members.

Supported Script Types

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.
N/task Module

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var workflowTask = task.create({taskType: task.TaskType.WORKFLOW_TRIGGER});
workflowTask.recordType = 'customer';
workflowTask.recordId = 107;
workflowTask.workflowId = 3;
var taskId = workflowTask.submit();
...
//Add additional code
```

WorkflowTriggerTask.submit()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Directs NetSuite to place the asynchronous workflow initiation task into the NetSuite scheduling queue and returns a unique ID for the task. Use WorkflowTriggerTaskStatus.status to view the status of a submitted task.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>the task id as a string</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server scripts  
For more information, see the help topic SuiteScript 2.0 Script Types.                                                                                     |
| Governance        | 20 units                                                                                                                                                                                        |
| Module            | N/task Module                                                                                                                                                                                  |
| Since             | 2015.2                                                                                                                                                                                          |

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_TO_SUBMIT_JOB_REQUEST_1</td>
<td>Failed to submit job request: {reason}</td>
<td>Task cannot be submitted.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var workflowTriggerTask = workflowTask.submit();
...
//Add additional code
```

WorkflowTriggerTask.recordType

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Record type of the workflow definition base record. For example, customer, salesorder, or lead.</th>
</tr>
</thead>
</table>

In the Workflow Manager, this is the record type that is specified in the Record Type field.

**Type**
string

**Supported Script Types**
Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/task Module

**Since**
2015.2

### WorkflowTriggerTask.recordId

**Property Description**
Internal ID of the base record. For example, 55 or 124.

**Type**
number

**Supported Script Types**
Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/task Module

**Since**
2015.2

### WorkflowTriggerTask.workflowId

**Property Description**
Internal ID (as a number), or script ID (as a string), for the workflow definition. This is the ID that appears in the ID field on the Workflow Definition Page.

**Type**
number | string
Supported Script Types
Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/task Module

Since
2015.2

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
workflowTask.workflowId = 3;
...
//Add additional code
```

**WorkflowTriggerTask.params**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Object that contains key/value pairs to set default values on fields specific to the workflow. These can include fields on the Workflow Definition Page or workflow and state Workflow Custom Fields.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Object</td>
</tr>
</tbody>
</table>
| Supported Script Types | Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types. |

Module
N/task Module

Since
2015.2

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
task.params = {context: portlet}
...
//Add additional code
```

task.WorkflowTriggerTaskStatus

**Object Description**
Encapsulates the status of an asynchronous workflow initiation task placed into the NetSuite task queue by WorkflowTriggerTask.submit().

Use task.checkStatus(options) with the unique ID for the asynchronous workflow initiation task to get the WorkflowTriggerTaskStatus object.
For a complete list of this object's properties, see WorkflowTriggerTaskStatus Object Members.

**Supported Script Types**
- Server scripts
  - For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
- N/task Module

**Since**
- 2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var workflowTaskStatus = task.checkStatus(taskId);
if (workflowTaskStatus.status === task.TaskStatus.FAILED)
...
//Add additional code
```

### WorkflowTriggerTaskStatus.status

**Property Description**
Status for an asynchronous workflow placed in the NetSuite task queue by WorkflowTriggerTask.submit(). Returns a task.TaskStatus enum value.

**Type**
task.TaskStatus

**Supported Script Types**
- Server scripts
  - For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**
- N/task Module

**Since**
- 2015.2

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
</table>
| READ_ONLY      | Setting the property is attempted

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var summary = task.checkStatus(scriptTaskId);
log.audit('Status', summary.status);
...
//Add additional code
```
task.RecordActionTask (Beta)

**Warning:** The record action task is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature's operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

**Object Description**
Encapsulates the properties of a record action task. Use the methods and properties for this object to submit a record action task into the task queue and to execute it asynchronously.

**Supported Script Types**
- Server scripts
  For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**
N/task Module

**Methods and Properties**
- [RecordActionTask (Beta) Object Members](#)

**Since**
2019.1

**Syntax**

```javascript
//Add additional code
...
{
  var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
  recordActionTask.recordType = 'timebill';
  recordActionTask.action = 'approve';
  recordActionTask.params = [{recordId: 1, note: "this is a note for 1"},
                   {recordId: 5, note: "this is a note for 5"},
                   {recordId: 23, note: "this is a note for 23"}];

  var handle = recordActionTask.submit();

  var res = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
  log.debug('Initial status: ' + res.status);
});
//Add additional code
```

**RecordActionTask.submit()**

**Note:** The record action task is a beta feature in 2019.1.

**Method Description**
Submits a record action task script deployment for processing and returns its task ID.
The record action task is processed by a background process which executes the specified record action for each record ID provided in the parameters. The overall task status as well as individual action results can be queried using the `task.checkStatus()` method.

**Returns**

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server scripts</td>
<td></td>
</tr>
</tbody>
</table>

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

<table>
<thead>
<tr>
<th>Module</th>
<th>50 units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sibling Object</td>
<td></td>
</tr>
<tr>
<td>Members</td>
<td></td>
</tr>
</tbody>
</table>

**Since**

2019.1

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILED_TO_SUBMIT_JOB_REQUEST_1</td>
<td>Failed to submit job request: {reason}</td>
<td>The task cannot be submitted.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
{
    var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
    recordActionTask.recordType = 'timebill';
    recordActionTask.action = 'approve';
    recordActionTask.params = [{recordId: 1, note: "this is a note for 1"},
                                {recordId: 5, note: "this is a note for 5"},
                                {recordId: 23, note: "this is a note for 23"}];

    var handle = recordActionTask.submit();

    var res = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
    log.debug('Initial status: ' + res.status);
});
//Add additional code
```

**RecordActionTask.toString()**

**Note:** The record action task is a beta feature in 2019.1.

**Method Description**

Returns the object type name.
Returns | string
---|---
Supported Script Types | Server scripts
Governance | None
Module | N/task Module
Sibling Object Members | RecordActionTask (Beta) Object Members
Since | 2019.1

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
log.debug("Task type: " + recordActionTask.toString());
// Add additional code
```

**RecordActionTask.toJSON()**

ℹ️ **Note:** The record action task is a beta feature in 2019.1.

Method Description | Returns an object in JSON.
---|---
Returns | Object
Supported Script Types | Server scripts
Governance | None
Module | N/task Module
Sibling Object Members | RecordActionTask (Beta) Object Members
Since | 2019.1

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
```
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}];
log.debug("Task details: " + recordActionTask.toJSON());
...
// Add additional code

RecordActionTask.paramCallback()

**Note:** The record action task is a beta feature in 2019.1.

**Description**
Property of type function that takes record ID and returns the parameter object for the specified record ID. Is to be used in conjunction with task.ActionCondition (Beta).

This parameter cannot be specified when RecordActionTask.params is specified.

**Supported Script Types**
Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/task Module

**Sibling Object Members**
RecordActionTask (Beta) Object Members

**Since**
2019.1

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.condition = task.ActionCondition.ALL_QUALIFIED_INSTANCES;
recordActionTask.paramCallback = function(v) {
    return { recordId: v, note: "this is a note for " + v };}
};
var handle = recordActionTask.submit();
...
// Add additional code
```

RecordActionTask.recordType

**Note:** The record action task is a beta feature in 2019.1.

**Parameter Description**
The record type on which the action is to be performed.
For a list of record types, see `record.Type`.

**Type**

string

**Supported Script Types**

Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/task Module

**Sibling Object Members**

RecordActionTask (Beta) Object Members

**Since**

2019.1

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
{
    var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
    recordActionTask.recordType = 'timebill';
    recordActionTask.action = 'approve';
    recordActionTask.params = [{recordId: 1, note: "this is a note for 1"},
                                {recordId: 5, note: "this is a note for 5"},
                                {recordId: 23, note: "this is a note for 23"}];

    var handle = recordActionTask.submit();

    var res = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
    log.debug('Initial status: ' + res.status);
}
...
//Add additional code
```

### RecordActionTask.action

**Note:** The record action task is a beta feature in 2019.1.

**Parameter Description**

The ID of the action to be invoked.

**Type**

string

**Supported Script Types**

Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/task Module

**Sibling Object Members**

RecordActionTask (Beta) Object Members

**Since**

2019.1
Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
{
  var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
  recordActionTask.recordType = 'timebill';
  recordActionTask.action = 'approve';
  recordActionTask.params = [{recordId: 1, note: "this is a note for 1"},
                             {recordId: 5, note: "this is a note for 5"},
                             {recordId: 23, note: "this is a note for 23"}];

  var handle = recordActionTask.submit();

  var res = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
  log.debug('Initial status: ' + res.status);
}
...;
// Add additional code
```

RecordActionTask.params

Note: The record action task is a beta feature in 2019.1.

Parameter Description | An array of parameter objects. Each object corresponds to one record ID of the record for which the action is to be executed. The object has the following form: `{recordId: 1, someParam: 'example1', otherParam: 'example2'}`
---|---
Type | Array of objects
Supported Script Types | Server scripts
For more information, see the help topic SuiteScript 2.0 Script Types.
Module | N/task Module
Sibling Object Members | RecordActionTask (Beta) Object Members
Since | 2019.1
RecordActionTask.condition

**Note:** The record action task is a beta feature in 2019.1.

**Parameter Description**
The condition used to select record IDs of records for which the action is to be executed.

This parameter is specified with the task.ActionCondition enum.

This is used in conjunction with RecordActionTask.paramCallback. If RecordActionTask.paramCallback is not specified, this default callback is used:

```javascript
function(v) { return { recordId: v }; }
```

<table>
<thead>
<tr>
<th>Type</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
<td></td>
</tr>
</tbody>
</table>

**Module**
N/task Module

**Sibling Object Members**
RecordActionTask (Beta) Object Members

**Since**
2019.1

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.condition = task.ActionCondition.ALL_QUALIFIED_INSTANCES;
recordActionTask.paramCallback = function(v) {
  return { recordId: v, note: "this is a note for " + v }; 
};
var handle = recordActionTask.submit();
...
// Add additional code
```
### task.RecordActionTaskStatus (Beta)

**Warning:** The record action task is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature's operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the properties of a record action task. Use the methods and properties for this object to submit a record action task into the task queue and to execute it asynchronously.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>Server scripts For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Methods and Properties</td>
<td>RecordActionTaskStatus (Beta) Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
// Add additional code...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 5}, {recordId: 23}]
var handle = recordActionTask.submit();

// Add any additional processing here

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug('Initial status: ' + taskStatus.status);

// Add additional code

/* Example contents of a RecordActionTaskStatus object at different stages of bulk action task execution:

// initial status just after submitting the task
{
  status: 'PENDING',
  results: {},
  errors: {},
  complete: 0,
  succeeded: 0,
  failed: 0,
  pending: 3
} */
```
// in the middle of processing, two records processed, one to go
{
  status: 'PROCESSING',
  results: {
    1: { response: { approvedId: 1 }, notifications: [] },
    5: { response: { approvedId: 5 }, notifications: [{ title: 'Title', message: 'Message', severity: { value: 2, label: 'Warning' } }] }
  },
  errors: {},
  complete: 2,
  succeeded: 2,
  failed: 0,
  pending: 1
}

// complete, all successful
{
  status: 'COMPLETE',
  results: {
    1: { response: { approvedId: 1 }, notifications: [] },
    23: { response: { approvedId: 23 }, notifications: [] }
  },
  errors: {},
  complete: 3,
  succeeded: 3,
  failed: 0,
  pending: 0
}

// complete, one action returned an error
{
  status: 'COMPLETE',
  results: {
    1: { response: { approvedId: 1 }, notifications: [] },
    23: { name: 'SSS_RECORD_DOES_NOT_SATISFY_CONDITION', message: '... ' }
  },
  errors: {
    23: { name: 'SSS_RECORD_DOES_NOT_SATISFY_CONDITION', message: '... ' }
  },
  complete: 3,
  succeeded: 2,
  failed: 1,
  pending: 0
}
*/

...
RecordActionTaskStatus.toString()

**Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns the object type name.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td>string</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/task Module</td>
</tr>
<tr>
<td><strong>Sibling Object Members</strong></td>
<td>RecordActionTaskStatus (Beta) Object Members</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2019.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug('Type of status object: ' + taskStatus.toString());
...
//Add additional code
```

RecordActionTaskStatus.toJSON()

**Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns a record status task status object in JSON.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td>Object</td>
</tr>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/task Module</td>
</tr>
<tr>
<td><strong>Sibling Object Members</strong></td>
<td>RecordActionTaskStatus (Beta) Object Members</td>
</tr>
</tbody>
</table>

SuiteScript 2.0 API Reference
Since 2019.1

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug('Status object details: ' + taskStatus.toJSON());
...
// Add additional code
```

RecordActionTaskStatus.status

⚠️ **Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>status</code></td>
<td>Represents the record action task status. Returns a value from the task.TaskStatus enum.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
</table>

Supported Script Types

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/task Module

Sibling Object Members

RecordActionTaskStatus (Beta) Object Members

Since 2019.1

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
```
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskIds: handle}); // returns a RecordActionTaskStatus object
log.debug('Current task status: ' + taskStatus.status);
// will log e.g. the following:
// Current task status: PENDING
...
// Add additional code

RecordActionTaskStatus.results

**Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>The results of successfully executed record action tasks. The value of the property is the task instance ID and the corresponding action result.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Type</strong></td>
<td>Object</td>
</tr>
</tbody>
</table>
| **Supported Script Types** | Server scripts  
For more information, see the help topic [SuiteScript 2.0 Script Types](#).
| **Module**            | N/task Module |
| **Sibling Object Members** | RecordActionTaskStatus (Beta) Object Members |
| **Since**             | 2019.1 |

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug(taskStatus.results);
// will log e.g. the following:
// { 1: { response: { approved: true }, notifications: [] }}
...
// Add additional code
```
RecordActionTaskStatus.errors

**Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The error details of failed action executions. The value of the property is the record instance ID and the corresponding error details. The error details are returned in an unnamed object with two properties: code and message.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Object</th>
</tr>
</thead>
</table>

**Supported Script Types**

Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/task Module

**Sibling Object Members**

RecordActionTaskStatus (Beta) Object Members

**Since**

2019.1

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug(taskStatus.errors);
// will log e.g. the following:
// { 2: { name: 'SSS_RECORD_DOES_NOT_SATISFY_CONDITION', message: '...' }}
...
// Add additional code
```

RecordActionTaskStatus.complete

**Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of record actions that are already executed, either failed or successful.</td>
<td></td>
</tr>
</tbody>
</table>
## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted.</td>
</tr>
</tbody>
</table>

## Syntax

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug('Actions already complete: ' + taskStatus.complete);
// will log e.g. the following:
// Actions already complete: 2
...
// Add additional code
```

## RecordActionTaskStatus.succeeded

### Note: The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>The number of record actions with a successful status.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>RecordActionTaskStatus (Beta) Object Members</td>
</tr>
</tbody>
</table>
### Since

2019.1

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug('Actions executed successfully: ' + taskStatus.succeeded);
// will log e.g. the following:
// Actions executed successfully: 1
...
// Add additional code
```

### RecordActionTaskStatus.failed

**Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The number of record actions with a failed status.</td>
</tr>
</tbody>
</table>

**Type**

number

**Supported Script Types**

Server scripts

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/task Module

**Sibling Object Members**

RecordActionTaskStatus (Beta) Object Members

**Since**

2019.1

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug('Actions failed: ' + taskStatus.failed);
// will log e.g. the following:
// Actions failed: 0
...
// Add additional code
```

RecordActionTaskStatus.pending

ℹ️ **Note:** The record action task is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Parameter Description</th>
<th>The number of record actions with a pending status.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Server scripts</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/task Module</td>
</tr>
<tr>
<td>Sibling Object Members</td>
<td>RecordActionTaskStatus (Beta) Object Members</td>
</tr>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>READ_ONLY</td>
<td>Setting the property is attempted.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
// Add additional code
...
var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.params = [{recordId: 1}, {recordId: 2}];
```
var handle = recordActionTask.submit();

var taskStatus = task.checkStatus({taskId: handle}); // returns a RecordActionTaskStatus object
log.debug('Actions pending: ' + taskStatus.pending);
// will log e.g. the following:
// Actions pending: 2
...
// Add additional code

### task.create(options)

| Method Description | Creates an object for a specific task type and returns the task object. Use with the N/task Module to create a task to schedule scripts, run map/reduce scripts, import CSV files, merge duplicate records, initiate asynchronous searches, or execute asynchronous workflows. |
| Returns | task.ScheduledScriptTask | task.MapReduceScriptTask | task.CsvImportTask | task.EntityDeduplicationTask | task.WorkflowTriggerTask | task.SearchTask |
| Supported Script Types | Server scripts |
| Governance | None |
| Module | N/task Module |
| Since | 2015.2 |

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.taskType</td>
<td>task.TaskType</td>
<td>Required</td>
<td>The type of task object to create. Use the task.TaskType enum to set the value.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.scriptId</td>
<td>number</td>
<td>Optional</td>
<td>The internal ID (as a number) or script ID (as a string) for the script record. This parameter sets the value for the ScheduledScriptTask.scriptId or MapReduceScriptTask.scriptId property. Only applicable when taskType is set to SCHEDULED_SCRIPT or MAP_REDUCE.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.deploymentId</td>
<td>number</td>
<td>Optional</td>
<td>The internal ID (as a number) or script ID (as a string) of the script deployment record. This parameter sets the value for the ScheduledScriptTask.deploymentId or MapReduceScriptTask.deploymentId property. Only applicable when taskType is set to SCHEDULED_SCRIPT or MAP_REDUCE.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>Optional</td>
<td>An object that represents key/value pairs that override static script parameter field values on the script deployment record.</td>
<td>2016.2</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------------------</td>
<td>-----------------</td>
<td>---------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>options.importFile</td>
<td>file.File</td>
<td>Optional</td>
<td>A CSV file to import. Use a file.File object or a string that represents the CSV text to be imported.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
<td>This parameter sets the value for the CsvImportTask.importFile property.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Only applicable when taskType is set to CSV_IMPORT.</td>
<td></td>
</tr>
<tr>
<td>options.mappingId</td>
<td>number</td>
<td>Optional</td>
<td>The internal ID (as a number) or script ID (as a string) of a saved import map that you created when you ran the Import Assistant. See task.CsvImportTask.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>string</td>
<td></td>
<td>This parameter sets the value for the CsvImportTask.mappingId property.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Only applicable when taskType is set to CSV_IMPORT.</td>
<td></td>
</tr>
<tr>
<td>options.queueId</td>
<td>number</td>
<td>Optional</td>
<td>Overrides the Queue Number property under Advanced Options on the Import Options page of the Import Assistant. Use this property to programmatically select an import queue and improve performance during the import.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This parameter sets the value for the CsvImportTask.queueId property.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Only applicable when taskType is set to CSV_IMPORT.</td>
<td></td>
</tr>
<tr>
<td>options.name</td>
<td>string</td>
<td>Optional</td>
<td>The name for the CSV import task.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>You can optionally set a different name for a scripted import task. In the UI, this name appears on the CSV Import Job Status page.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This parameter sets the value for the CsvImportTask.name property.</td>
<td></td>
</tr>
</tbody>
</table>

Use these parameters for the task object to programmatically pass values to the script deployment. For more information about script parameters, see the help topic Creating Script Parameters Overview.

For Workflow tasks, keys can include fields on the Workflow Definition Page or workflow and state Workflow Custom Fields.

This parameter sets the value for the ScheduledScriptTask.params, MapReduceScriptTask.params or WorkflowTriggerTask.params property.

Only applicable when taskType is set to SCHEDULED_SCRIPT, MAP_REDUCE or WORKFLOW_TRIGGER.

**Note:** This property is only available if you have a SuiteCloud Plus license. For more information about using multiple queues when importing CSV files, see the help topics Queue Number and Use Multiple Threads and Multiple Queues to Run CSV Import Jobs.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.linkedFiles</td>
<td>Object</td>
<td>Optional</td>
<td>A map of key/value pairs that sets the data to be imported in a linked file for a multi-file import job, by referencing a file in the file cabinet or the raw CSV data to import. The key is the internal ID of the record sublist for which data is being imported and the value is either a file.File object or the raw CSV data to import. You can assign multiple types of values to the linkedFiles property. This parameter sets the value for the CsvImportTask.linkedFiles property. Only applicable when taskType is set to CSV_IMPORT.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.entityType</td>
<td>string</td>
<td>Optional</td>
<td>Sets the type of entity on which you want to merge duplicate records. This parameter sets the value for the EntityDeduplicationTask.entityType property. Only applicable when taskType is set to ENTITY_DEDUPICATION. Use the task.DedupeEntityType enum to set the value.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.masterRecordId</td>
<td>number</td>
<td>Optional</td>
<td>When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record. Use this property to set the ID of the master record that you want to use as the master record in the merge. This parameter sets the value for the EntityDeduplicationTask.masterRecordId property. Only applicable when taskType is set to ENTITY_DEDUPICATION.</td>
<td>2016.2</td>
</tr>
<tr>
<td>options.masterSelectionMode</td>
<td>string</td>
<td>Optional</td>
<td>When you merge duplicate records, you can delete all duplicates for a record or merge information from the duplicate records into the master record. Set this property to determine which of the duplicate records to keep or select the master record to use by ID. This parameter sets the value for the EntityDeduplicationTask.masterSelectionMode property. Only applicable when taskType is set to ENTITY_DEDUPICATION.</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

**Note:** If you set entityType to CUSTOMER, the system will automatically include prospects and leads in the task request.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>masterSelectionMode</code></td>
<td></td>
<td>Optional</td>
<td>This parameter sets the value for the <code>EntityDeduplicationTask.masterSelectionMode</code> property. Only applicable when <code>taskType</code> is set to <code>ENTITY_DEDUPLICATION</code>. Use the <code>task.MasterSelectionMode</code> enum to set the value.</td>
<td>2016.2</td>
</tr>
<tr>
<td><code>dedupeMode</code></td>
<td><code>string</code></td>
<td>Optional</td>
<td>Sets the mode in which to merge or delete duplicate records. This parameter sets the value for the <code>EntityDeduplicationTask.dedupeMode</code> property. Only applicable when <code>taskType</code> is set to <code>ENTITY_DEDUPLICATION</code>. Use the <code>task.DedupeMode</code> enum to set the value.</td>
<td>2016.2</td>
</tr>
<tr>
<td><code>recordIds</code></td>
<td><code>number[]</code></td>
<td>Optional</td>
<td>The number array of record internal IDs to perform the merge or delete operation on. You can use the <code>search.duplicates(options)</code> method to identify duplicate records or create an array with record internal IDs. This parameter sets the value for the <code>EntityDeduplicationTask.recordIds</code> property. Only applicable when <code>taskType</code> is set to <code>ENTITY_DEDUPLICATION</code>.</td>
<td>2016.2</td>
</tr>
<tr>
<td><code>recordType</code></td>
<td><code>string</code></td>
<td>Optional</td>
<td>The record type of the workflow definition base record, such as customer, salesorder, or lead. In the Workflow Manager, this is the record type that is specified in the Record Type field. This parameter sets the value for the <code>WorkflowTriggerTask.recordType</code> property. Only applicable when <code>taskType</code> is set to <code>WORKFLOW_TRIGGER</code>.</td>
<td>2016.2</td>
</tr>
<tr>
<td><code>recordId</code></td>
<td><code>number</code></td>
<td>Optional</td>
<td>The internal ID of the base record. This parameter sets the value for the <code>WorkflowTriggerTask.recordId</code> property. Only applicable when <code>taskType</code> is set to <code>WORKFLOW_TRIGGER</code>.</td>
<td>2016.2</td>
</tr>
<tr>
<td><code>workflowId</code></td>
<td><code>number</code></td>
<td>Optional</td>
<td>The internal ID (as a number) or script ID (as a string) for the workflow definition. This is the ID that appears in the ID field on the Workflow Definition Page. This parameter sets the value for the <code>WorkflowTriggerTask.workflowId</code> property. Only applicable when <code>taskType</code> is set to <code>WORKFLOW_TRIGGER</code>.</td>
<td>2016.2</td>
</tr>
<tr>
<td><code>savedSearchId</code></td>
<td><code>number</code></td>
<td>Optional</td>
<td>The ID of the saved search to be executed during the task.</td>
<td>2017.1</td>
</tr>
<tr>
<td><code>fileId</code></td>
<td><code>string</code></td>
<td>Optional</td>
<td>The ID of the CSV file to export search results to. See N/file Module.</td>
<td>2017.1</td>
</tr>
</tbody>
</table>
Parameter | Type | Required / Optional | Description |
--- | --- | --- | --- |
options.filePath | number | Optional | Path of the CSV file to export search results to. See N/file Module. |

**Note:** If fileId is provided then the filePath parameter is ignored. There is no synchronization between fileId and filePath values.

---

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
var mrTask = task.create({
  taskType: task.TaskType.MAP_REDUCE,
  scriptId: 34,
  deploymentId: 1,
  params: {doSomething: true}
});
...
//Add additional code
```

**task.checkStatus(options)**

**Method Description**

Returns a task status object associated with a specific task ID.

**Returns**

- task.ScheduledScriptTaskStatus
- task.MapReduceScriptTaskStatus
- task.CsvImportTaskStatus
- task.EntityDeduplicationTaskStatus
- task.SearchTaskStatus
- task.WorkflowTriggerTaskStatus

**Supported Script Types**

- Server scripts
  
  For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

- None

**Module**

- N/task Module

**Since**

- 2015.2
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>optionstaskId</td>
<td>task.ScheduledScriptTask</td>
<td>Required</td>
<td>Unique ID for the task that was generated by task.create(options).</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var taskStatus = task.checkStatus(mrTaskId);
...
//Add additional code
```

**task.TaskType**

**Enum Description**

Enumeration that holds the string values for the types of task objects supported by the N/task Module, that you can create with task.create(options).

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

Server scripts

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/task Module

**Since**

2015.2

**Values**

- SCHEDULED_SCRIPT
- MAP_REDUCE
- CSV_IMPORT
- ENTITY_DEDUPICATION
- SEARCH
- WORKFLOW_TRIGGER
## RECORD_ACTION

**Note:** The record action task is a beta feature in 2019.1

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
//Add additional code
...
var mrTask = task.create({
    taskType: task.TaskType.MAP_REDUCE
});
...
//Add additional code
```

### task.TaskStatus

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for the possible status of tasks created and submitted with the N/task Module.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The following properties hold a value for task.taskStatus:</td>
</tr>
<tr>
<td></td>
<td>- ScheduledScriptTaskStatus.status</td>
</tr>
<tr>
<td></td>
<td>- MapReduceScriptTaskStatus.status</td>
</tr>
<tr>
<td></td>
<td>- CsvImportTaskStatus.status</td>
</tr>
<tr>
<td></td>
<td>- EntityDeduplicationTaskStatus.status</td>
</tr>
<tr>
<td></td>
<td>- SearchTaskStatus.status</td>
</tr>
<tr>
<td></td>
<td>- WorkflowTriggerTaskStatus.status</td>
</tr>
</tbody>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/task Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Values

- PENDING
- PROCESSING
- COMPLETE
- FAILED
Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
if (status == task.TaskStatus.COMPLETE || status == task.TaskStatus.FAILED)
...
//Add additional code
```

task.MasterSelectionMode

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration that holds the string values for supported master selection modes when merging duplicate records with task.EntityDeduplicationTask.</td>
<td></td>
</tr>
</tbody>
</table>

Use this enum for the EntityDeduplicationTask.masterSelectionMode property.

For more information about these values, see the help topic Merging or Deleting Duplicate Records.

Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server scripts</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/task Module</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

Values

- CREATED_EARLIEST
- MOST_RECENT_ACTIVITY
- MOST_POPULATED_FIELDS
- SELECT_BY_ID

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/task Module Script Samples.

```javascript
//Add additional code
...
dedupeTask.masterSelectionMode = task.MasterSelectionMode.MOST_RECENT_ACTIVITY;
...
//Add additional code
```
task.DedupeMode

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for the available deduplication modes when merging duplicate records with task.EntityDeduplicationTask. Use this enum for the EntityDeduplicationTask.dedupeMode property.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

- **VALUES**
  - MERGE
  - DELETE
  - MAKE_MASTER_PARENT
  - MARK_AS_NOT_DUPES

**Syntax**

```javascript
//Add additional code
...
dedupeTask.dedupeMode = task.DedupeMode.MERGE;
...
//Add additional code
```

**Supported Script Types**

- Server scripts
  - For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

- N/task Module

**Since**

- 2015.2

---

**task.DedupeEntityType**

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for entity types for which you can merge duplicate records with task.EntityDeduplicationTask. Use this enum for the EntityDeduplicationTask.entityType.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong></td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

**Supported Script Types**

- Server scripts
  - For more information, see the help topic SuiteScript 2.0 Script Types.
**N/task Module**

<table>
<thead>
<tr>
<th>Module</th>
<th>N/task Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Values**

- CUSTOMER
- CONTACT
- VENDOR
- PARTNER
- LEAD
- PROSPECT

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see **N/task Module Script Samples**.

```javascript
//Add additional code
...
dedupeTask.entityType = task.DedupeEntityType.CUSTOMER;
...
//Add additional code
```

---

**task.MapReduceStage**

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for possible stages in <code>task.MapReduceScriptTask</code> for a map/reduce script.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This enum is returned by <code>MapReduceScriptTaskStatus.stage</code>. For general information about map/reduce stages, see the help topics <code>Map/Reduce Key Concepts</code> and <code>SuiteScript 2.0 Map/Reduce Script Stages</code>.</td>
</tr>
</tbody>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

- Server scripts
- For more information, see the help topic **SuiteScript 2.0 Script Types**.

<table>
<thead>
<tr>
<th>Module</th>
<th>N/task Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Values**

- GET_INPUT
- MAP
- SHUFFLE
## REDUCE
- **SUMMARIZE**

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
// Add additional code
...
if (summary.stage === task.MapReduceStage.SUMMARIZE)
...
// Add additional code
```

**Note:** For general information about map/reduce scripts, see the help topic [SuiteScript 2.0 Map/Reduce Script Type](#).

### task.ActionCondition (Beta)

**Warning:** The record action task is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature’s operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration that holds the string values for the possible record action conditions. This enum is returned by <code>RecordActionTask.condition</code>.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Server scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>For more information, see the help topic <a href="#">SuiteScript 2.0 Script Types</a>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/task Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2019.1</td>
</tr>
</tbody>
</table>

### Values

- **ALL_QUALIFIED_INSTANCES**

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/task Module Script Samples](#).

```javascript
// Add additional code
```
... var recordActionTask = task.create({taskType: task.TaskType.RECORD_ACTION});
recordActionTask.recordType = 'timebill';
recordActionTask.action = 'approve';
recordActionTask.condition = task.ActionCondition.ALL_QUALIFIED_INSTANCES;
recordActionTask.paramCallback = function(v) {
    return { recordId: v, note: 'this is a note for ' + v };}
};
var handle = recordActionTask.submit();
...
// Add additional code

N/transaction Module

Load the transaction module to void transactions.

When you void a transaction, the total and all the line items for the transaction are set to zero. The transaction is not removed from the system. NetSuite supports two types of voids: direct voids and voids by reversing journal. For additional information, see the help topic Voiding, Deleting, or Closing Transactions.

The type of void performed with your script depends on the targeted account’s preference settings:

- If the Using Reversing Journals preference is disabled, a direct void is performed.
- If the Using Reversing Journals preference is enabled, a void by reversing journal is performed.

⚠️ Important: After you successfully void a transaction, you can no longer make changes to the transaction that impact the general ledger.

- N/transaction Module Members
- N/transaction Module Script Sample

N/transaction Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>transaction.###void(options)</td>
<td>number</td>
<td>Client and server-side scripts</td>
<td>Voids a transaction record.</td>
</tr>
<tr>
<td></td>
<td>transaction.###void.promise(options)</td>
<td>number</td>
<td>Client scripts</td>
<td>Voids a transaction record asynchronously.</td>
</tr>
<tr>
<td>Enum</td>
<td>transaction.Type</td>
<td>enum</td>
<td>Client and server-side scripts</td>
<td>Enumeration that holds the string values for supported record types.</td>
</tr>
</tbody>
</table>

N/transaction Module Script Sample

⚠️ Note: This sample script uses the require function so that you can copy it into the debugger and test it. Keep in mind that you must use the define function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.
The following examples voids a transaction.

**Warning:** The following sample works only in OneWorld accounts.

```javascript
/**
 * @NApiVersion 2.x
 */
// This example shows how to void a transaction
require(['N/transaction', 'N/config', 'N/record'],
    function(transaction, config, record) {
        function voidSalesOrder() {
            var accountingConfig = config.load({
                type: config.Type.ACCOUNTING_PREFERENCES
            });
            accountingConfig.setValue({
                fieldId: 'REVERSALVOIDING',
                value: false
            });
            accountingConfig.save();
            var salesOrderObj = record.create({
                type: 'salesorder',
                isDynamic: false
            });
            salesOrderObj.setValue({
                fieldId: 'entity',
                value: 107
            });
            salesOrderObj.setSublistValue({
                sublistId: 'item',
                fieldId: 'item',
                value: 233,
                line: 0
            });
            salesOrderObj.setSublistValue({
                sublistId: 'item',
                fieldId: 'amount',
                value: 1,
                line: 0
            });
            var salesOrderId = salesOrderObj.save();
            var voidSalesOrderId = transaction.void({
                type: record.Type.SALES_ORDER,
                id: salesOrderId
            });
            var salesOrder = record.load({
                type: 'salesorder',
                id: voidSalesOrderId
            });
            // memo should be 'VOID'
            var memo = salesOrder.getValue({
                fieldId: 'memo'
            });
            voidSalesOrder();
        }
        voidSalesOrder();
    });
```
transaction.void(options)

Method Description
Method used to void a transaction record object and return an id that indicates the type of void performed.
The type of void performed depends on the targeted account's preference settings.

⚠️ Important: After you void a transaction, you cannot make changes to the transaction that impact the general ledger.

Returns
An ID returned as a number.
- If a direct void is performed, returns the ID of the record voided.
- If a void by reversing journal is performed, returns the ID of the newly created voiding journal.

Supported Script Types
All client and server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
10 units
Module
N/transaction Module
Since
2015.2

Parameters

⚠️ Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>number</td>
<td>string</td>
<td>required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Internal ID of the specific transaction</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>record instance to void.</td>
<td></td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td></td>
<td>required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Internal ID of the type of transaction</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>record to void</td>
<td></td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVALID_RECORD_TYPE</td>
<td>The type argument passed is not valid or the record type is not voidable.</td>
<td></td>
</tr>
<tr>
<td>THAT_RECORD_DOES_NOT_EXIST</td>
<td>The id argument passed is not valid.</td>
<td></td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>The type or id argument is missing.</td>
<td></td>
</tr>
</tbody>
</table>
### transaction.void.promise(options)

**Method Description**
Method used to void a transaction record object asynchronously and return an id that indicates the type of void performed.

The type of void performed depends on the targeted account’s preference settings.

**Important:** After you void a transaction, you cannot make changes to the transaction that impact the general ledger.

**Note:** For information about the parameters and errors thrown for this method, see `transaction.void(options)`. For additional information on promises, see Promise Object.

**Returns**
An id returned as a number.
- If a direct void is performed, returns the ID of the record voided.
- If a void by reversing journal is performed, returns the ID of the newly created voiding journal.

**Synchronous Version**
`transaction.void(options)`

**Supported Script Types**
All client-side scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**
10 units

**Module**
N/transaction Module

**Since**
2015.2

### Syntax

```
// Add additional code
...
var voidSalesOrderId = transaction.void.promise({
    type: record.Type.SALES_ORDER,
    id: salesOrderId
});
...
// Add additional code
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete promise script example, see Promise Object.
### transaction.Type

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Enumeration that holds the string values for supported transaction record types.</td>
</tr>
<tr>
<td><strong>Note:</strong></td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All client and server-side scripts</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/transaction Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Values

<table>
<thead>
<tr>
<th>Transaction Record</th>
<th>Supported Void Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASSEMBLY_BUILD</td>
<td>None</td>
</tr>
<tr>
<td>ASSEMBLY_UNBUILD</td>
<td>None</td>
</tr>
<tr>
<td>BIN_TRANSFER</td>
<td>None</td>
</tr>
<tr>
<td>BIN_WORKSHEET</td>
<td>None</td>
</tr>
<tr>
<td>BLANKET_PURCHASE_ORDER</td>
<td>None</td>
</tr>
<tr>
<td>CASH_REFUND</td>
<td>Direct Void</td>
</tr>
<tr>
<td>CASH_SALE</td>
<td>Direct Void</td>
</tr>
<tr>
<td>CHECK</td>
<td>Void by Reversing Journal</td>
</tr>
<tr>
<td>CREDIT_CARD_CHARGE</td>
<td>None</td>
</tr>
<tr>
<td>CREDIT_CARD_REFUND</td>
<td>None</td>
</tr>
<tr>
<td>CREDIT_MEMO</td>
<td>Direct Void</td>
</tr>
<tr>
<td>CUSTOMER_DEPOSIT</td>
<td>Direct Void</td>
</tr>
<tr>
<td>CUSTOMER_PAYMENT</td>
<td>Direct Void</td>
</tr>
<tr>
<td>CUSTOMER_PAYMENT_AUTHORIZATION</td>
<td>None</td>
</tr>
<tr>
<td>CUSTOMER_REFUND</td>
<td>Direct Void and Void by Reversing Journal</td>
</tr>
<tr>
<td>CUSTOM.Transaction</td>
<td>None</td>
</tr>
<tr>
<td>DEPOSIT</td>
<td>None</td>
</tr>
<tr>
<td>DEPOSIT_APPLICATION</td>
<td>None</td>
</tr>
<tr>
<td>Transaction Record</td>
<td>Supported Void Type</td>
</tr>
<tr>
<td>------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>ESTIMATE</td>
<td>Direct Void</td>
</tr>
<tr>
<td>EXPENSE_REPORT</td>
<td>Direct Void</td>
</tr>
<tr>
<td>FULFILLMENT_REQUEST</td>
<td>None</td>
</tr>
<tr>
<td>INBOUND_SHIPMENT</td>
<td>None</td>
</tr>
<tr>
<td>INVENTORY_ADJUSTMENT</td>
<td>None</td>
</tr>
<tr>
<td>INVENTORY_COST_REVALUATION</td>
<td>None</td>
</tr>
<tr>
<td>INVENTORY_COUNT</td>
<td>None</td>
</tr>
<tr>
<td>INVENTORY_STATUS_CHANGE</td>
<td>None</td>
</tr>
<tr>
<td>INVENTORY_TRANSFER</td>
<td>None</td>
</tr>
<tr>
<td>INVOICE</td>
<td>DirectVoid</td>
</tr>
<tr>
<td>ITEM_FULFILLMENT</td>
<td>None</td>
</tr>
<tr>
<td>ITEM_RECEIPT</td>
<td>None</td>
</tr>
<tr>
<td>JOURNAL_ENTRY</td>
<td>Direct Void</td>
</tr>
<tr>
<td>OPPORTUNITY</td>
<td>None</td>
</tr>
<tr>
<td>PAYCHECK</td>
<td>None</td>
</tr>
<tr>
<td>PAYCHECK_JOURNAL</td>
<td>Direct Void</td>
</tr>
<tr>
<td>PERIOD_END_JOURNAL</td>
<td>None</td>
</tr>
<tr>
<td>PURCHASE_CONTRACT</td>
<td>None</td>
</tr>
<tr>
<td>PURCHASE_ORDER</td>
<td>None</td>
</tr>
<tr>
<td>PURCHASE_REQUISITION</td>
<td>None</td>
</tr>
<tr>
<td>RETURN_AUTHORIZATION</td>
<td>Direct Void</td>
</tr>
<tr>
<td>REVENUE_ARRANGEMENT</td>
<td>None</td>
</tr>
<tr>
<td>REVENUE_COMMITMENT</td>
<td>None</td>
</tr>
<tr>
<td>REVENUE_COMMITMENT_REVERSAL</td>
<td>None</td>
</tr>
<tr>
<td>SALES_ORDER</td>
<td>Direct Void</td>
</tr>
<tr>
<td>STORE_PICKUP_FULFILLMENT</td>
<td>None</td>
</tr>
<tr>
<td>TRANSFER_ORDER</td>
<td>Direct Void</td>
</tr>
<tr>
<td>VENDOR_BILL</td>
<td>Direct Void</td>
</tr>
<tr>
<td>VENDOR_CREDIT</td>
<td>Direct Void</td>
</tr>
<tr>
<td>VENDOR_PAYMENT</td>
<td>Direct Void and Void by Reversing Journal</td>
</tr>
<tr>
<td>VENDOR_RETURN_AUTHORIZATION</td>
<td>Direct Void</td>
</tr>
<tr>
<td>WORK_ORDER</td>
<td>Direct Void</td>
</tr>
</tbody>
</table>
### Support Void Type

<table>
<thead>
<tr>
<th>Transaction Record</th>
<th>Supported Void Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK_ORDER_CLOSE</td>
<td>Direct Void</td>
</tr>
<tr>
<td>WORK_ORDER_COMPLETION</td>
<td>Direct Void</td>
</tr>
<tr>
<td>WORK_ORDER_ISSUE</td>
<td>Direct Void</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/record Module Script Samples.

```plaintext
//Add additional code
...
var voidSalesOrderId = transaction.void({
  type: transaction.Type.SALES_ORDER,
  id: salesOrderId
});
...
//Add additional code
```

---

### N/translation Module

**Warning:** The N/translation module is a beta feature. The contents of this feature are preliminary and may be changed or discontinued without prior notice. Any change may impact the feature’s operation with the NetSuite application. Warranties and product service levels do not apply to this feature or the impact of the feature on other portions of the NetSuite application. We may review and monitor the performance and use of this feature. The documentation for this feature is also considered a beta version and is subject to revision.

The N/translation module lets SuiteScript developers interact with NetSuite Translation Collections programmatically. The N/translation module is a SuiteScript 2.0 module, and both the N/translation module and Translation Collections are beta features in 2019.1. For more information about Translation Collections, see the help topic Translation Collection Overview.

A Translation Collection is a customization object that stores translation terms with their translations. A single Translation Collection can contain up to 1,000 translation terms. A translation term is a key/value pair where the key is an identifier and its value is a translatable string. A key references one string that can be translated into multiple languages. For example, a translation term for the word “hello” could consist of a key called HELLO and a string value of “hello”. You can translate a string into any language supported by NetSuite. For a list of these languages, see the help topic Configuring Multiple Languages.

You can create a collection of terms for translation in the NetSuite UI. To create this collection, your role must have the Manage Translations permission, or you must be using an Administrator role. You can export the collection of terms as an XLIFF translation file with a .xlf extension and send this file to a translation vendor. After the translation vendor translates the collection of terms, you can import the translation file back into your NetSuite account. You can use the collection of terms to translate labels and messages in the NetSuite UI and in SuiteApps. For information about managing Translation Collections in the UI, see the help topic About the Manage Translations Page.

You can use the N/translation module to access the translation terms stored in Translation Collections. The N/translation module provides read-only access to Translation Collections.
are managed in the NetSuite UI, and you cannot create or modify Translation Collections using SuiteScript.

A Translation Collection is encapsulated in the translation.Handle object. The translation.Handle object is a hierarchical object, which means that each node in the object is either another translation.Handle object or a translation.Translator function. Translator functions combine strings with parameters. When you create a Translation Collection in the NetSuite UI, you can include parameter placeholders in your translation strings. The translator function injects the specified parameter values into the placeholders in the returned translation string.

In your scripts, use translation.get(options) to get a translation.Translator function you can use to obtain specific translated strings in a collection. Consider the following code sample:

```javascript
// key HELLO_1 = 'Hello, {1}'

message: translation.get({
  collection: 'custcollection_my_strings',
  key: 'HELLO_1'
})(
  params: 'NetSuite'
)}
```

In this sample, if the string value of the HELLO_1 key is "Hello, {1}", the translation.Translator function combines the string with the params parameter value and returns "Hello, NetSuite". You can also use translation.load(options) to load translation terms from one or more Translation Collections. For information about the way strings are added to and formatted in collections, see the help topic Working with Translation Collection Strings.

You can load collections in different language locales by using the locales parameter of translation.load(options). You can also use translation.selectLocale(options) to create a translation.Handle object in a specific locale from an existing translation.Handle object.

- N/translation Module Members
- N/translation Module Script Samples

### N/translation Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>translation.Handle</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates a Translation Collection for a locale.</td>
</tr>
<tr>
<td></td>
<td>translation.Translator</td>
<td>Object / Function</td>
<td>Client and server-side scripts</td>
<td>Represents a translator function that returns translated strings. The translated strings include variables that are passed as parameters to the translator function.</td>
</tr>
</tbody>
</table>

### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>translation.#get(options)</td>
<td>Creates a translator function for a key in the specified Translation Collection and locale.</td>
</tr>
<tr>
<td>translation.#load(options)</td>
<td>Creates a translation.Handle object with translations for the specified Translation Collections and locales.</td>
</tr>
<tr>
<td>translation.#selectLocale(options)</td>
<td>Creates a translation.Handle object in the specified locale from an existing translation.Handle object.</td>
</tr>
</tbody>
</table>
**N/translation Module Script Samples**

See the following script samples for examples of how to use the N/translation module.

- Sample 1 – Get a translation string
- Sample 2 – Get a localized translation string
- Sample 3 – Get a translation string with parameters
- Sample 4 – Load specific translation strings from a collection
- Sample 5 – Load translation strings from multiple collections
- Sample 6 – Load a Translation Collection with multiple locales

**Sample 1 – Get a translation string**

The following script sample accesses translation strings one at a time using `translation.get(options)`. This method returns a translator function, which is subsequently called with any specified parameters. The translator function returns the string in the user’s session locale by default.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/ui/message', 'N/translation'],
function(message, translation) {

  // Create a message with translated strings
  var myMsg = message.create({
    title: translation.get({
      collection: 'custcollection_my_strings',
      key: 'MY_TITLE'
    })(),
    message: translation.get({
      collection: 'custcollection_my_strings',
      key: 'MY_MESSAGE'
    })(),
    type: message.Type.CONFIRMATION
  });

  // Show the message for 5 seconds
  myMsg.show({
    duration: 5000
  });
});
```
Sample 2 – Get a localized translation string

The following script sample accesses translation strings using a locale other than the default locale. When you call `translation.get(options)` and do not specify a locale, the method uses the current user's session locale. You can use the `options.locale` parameter to specify another locale. The `translation.Locale` enum lists all locales that are enabled for a company, and you can use these locales in `translation.get(options)`. The `translation.Locale` enum also includes two special values: `CURRENT` and `COMPANY_DEFAULT`. The `CURRENT` value represents the current user’s locale, and the `COMPANY_DEFAULT` value represents the default locale for the company.

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

require([ 'N/ui/message', 'N/translation' ],
function(message, translation) {

    // Create a message with translated strings
    var myMsg = message.create({
        title: translation.get({
            collection: 'custcollection_my_strings',
            key: 'MY_TITLE',
            locale: translation.Locale.COMPANY_DEFAULT
        })(),
        message: translation.get({
            collection: 'custcollection_my_strings',
            key: 'MY_MESSAGE',
            locale: translation.Locale.COMPANY_DEFAULT
        })(),
        type: message.Type.CONFIRMATION
    });

    // Show the message for 5 seconds
    myMsg.show({
        duration: 5000
    });
});
```

Sample 3 – Get a translation string with parameters

The following script sample accesses parametrized translation strings. When you create a Translation Collection in the NetSuite UI, you can include parameter placeholders in your translation strings. Placeholders use braces and a number (starting from 1). The translator function injects the specified parameter values into the placeholders in the translation string. For example, “Hello, {1}!” is a valid translation string, where {1} is a placeholder for a parameter. In this script sample, the parameter “NetSuite” is provided to the translator function returned from `translation.get(options)`, and the translator function returns a translated string of “Hello, NetSuite!”

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

require([ 'N/ui/message', 'N/translation' ],
function(message, translation) {

    // Create a message with translated strings
    var myMsg = message.create({
        title: translation.get({
            collection: 'custcollection_my_strings',
            key: 'MY_TITLE',
            locale: translation.Locale.COMPANY_DEFAULT
        })(),
        message: translation.get({
            collection: 'custcollection_my_strings',
            key: 'MY_MESSAGE',
            locale: translation.Locale.COMPANY_DEFAULT
        })(),
        type: message.Type.CONFIRMATION
    });

    // Show the message for 5 seconds
    myMsg.show({
        duration: 5000
    });
});
```
Sample 4 – Load specific translation strings from a collection

The following script sample loads specific translation strings from a collection. The translation.load(options) method can load a maximum of 1,000 translation strings. If you need only a few of the translation strings in a collection, you can load only the strings you need instead of loading the entire collection.

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

require(['N/ui/message', 'N/translation'],
function(message, translation) {

  // Load translation strings by key
  var localizedStrings = translation.load({
    collections: [{
      alias: 'myCollection',
      collection: 'custcollection_my_strings',
      keys: ['MY_TITLE', 'MY_MESSAGE']
    }
  ]);,

  // Create a message with translated strings
  var myMsg = message.create({
    title: localizedStrings.myCollection.MY_TITLE(),
    message: localizedStrings.myCollection.MY_MESSAGE(),
    type: message.Type.CONFIRMATION
  });

  // Show the message for 5 seconds
  myMsg.show({
    duration: 5000
  });
});
```
Sample 5 – Load translation strings from multiple collections

The following script sample loads translation strings from multiple Translation Collections in a single call of translation.load(options). This method can load a maximum of 1,000 translation strings, regardless of whether the strings are loaded from one collection or multiple collections.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/ui/message', 'N/translation'],
function(message, translation) {

    // Load two Translation Collections
    var localizedStrings = translation.load(
    {
        collections: [{
            alias: 'myCollection',
            collection: 'custcollection_my_strings'
        },
        {
            alias: 'myOtherCollection',
            collection: 'custcollection_other_strings'
        }
    }
    );

    // Create a message with translated strings
    var myMsg = message.create({
        title: localizedStrings.myCollection.MY_TITLE(),
        message: localizedStrings.myOtherCollection.MY_OTHER_MESSAGE(),
        type: message.Type.CONFIRMATION
    });

    // Show the message for 5 seconds
    myMsg.show({
        duration: 5000
    });
});
```

Sample 6 – Load a Translation Collection with multiple locales

The following script sample loads a Translation Collection with multiple locales. When you load translation strings using translation.load(options), you can specify a list of valid locales for the strings. You can use these locales when you select a locale using translation.selectLocale(options). If you specify more than one locale when you call translation.load(options), the first specified locale in the list is used for the created translation.Handle object. If you want to use a different locale from the list, use translation.selectLocale(options), which returns a translation.Handle object in the specified locale. You must load a locale using translation.load(options) before you can select it using translation.selectLocale(options).

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/ui/message', 'N/translation'],
```
function(message, translation) {

    // Load a Translation Collection and a set of locales
    var germanStrings = translation.load({
        collections: [{
            alias: 'myCollection',
            collection: 'custcollection_my_strings'
        }],
        locales: [translation.Locale.de_DE, translation.Locale.es_ES]
    });

    // Select a locale from the list of loaded locales
    var spanishStrings = translation.selectLocale({
        handle: germanStrings,
        locale: translation.Locale.es_ES
    });

    // Create a message with translated strings
    var myMsg = message.create({
        title: germanStrings.myCollection.MY_TITLE(),
        message: spanishStrings.myCollection.MY_MESSAGE(),
        type: message.Type.CONFIRMATION
    });

    // Show the message for 5 seconds
    myMsg.show({
        duration: 5000
    });
};

translation.Handle

Note: The N/translation module is a beta feature in 2019.1.

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates a Translation Collection for a locale.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use <strong>translation.load(options)</strong> to create a <strong>translation.Handle</strong> object with translations for the specified Translation Collections and locales. Use <strong>translation.selectLocale(options)</strong> to create a <strong>translation.Handle</strong> object in the specified locale from an existing <strong>translation.Handle</strong> object.</td>
<td></td>
</tr>
<tr>
<td>The <strong>translation.Handle</strong> object is a hierarchical object, which means that each node in the object is either another <strong>translation.Handle</strong> object or a <strong>translation.Translator</strong> function. Translator functions combine strings with parameters. When you create a Translation Collection in the NetSuite UI, you can include parameter placeholders in your translation strings. The translator function injects the specified parameter values into the placeholders in the returned translation string.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>Client and server-side scripts</th>
</tr>
</thead>
<tbody>
<tr>
<td>For additional information, see the help topic <strong>SuiteScript 2.0 Script Types</strong>.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/translation Module</th>
</tr>
</thead>
</table>

| Sibling Object Members | N/translation Module Members |
The N/translation module is a beta feature in 2019.1.

**Object / Function Description**

- Represents a translator function that returns translated strings.

**Note:** The N/translation module is a beta feature in 2019.1.

Use `translation.get(options)` to obtain this function for the specified Translation Collection and locale. The translator function is called with any parameters that you specify, and the translator function returns the appropriate translated string.

When you create a Translation Collection in the NetSuite UI, you can include parameter placeholders in your translation strings. Translation strings that include placeholders are called parametrized translation strings. Placeholders use braces and a number (starting from 1). The translator function injects the specified parameter values into the placeholders in the translation string.

For example, “Hello, {1}!” is a valid translation string, where {1} is a placeholder for a parameter. If you call `translation.get(options)` and specify a parameter of “NetSuite”, the translator function returns “Hello, NetSuite!” in the appropriate locale.

**Supported Script Types**

Client and server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.params</td>
<td>string[]</td>
<td>optional</td>
<td>The parameters to pass to the translator function. The parameter values are used in parametrized translation strings.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>The function parameters were not passed as an array.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
// Add additional code
...
var myMsg = message.create({
  title: translation.get({
    collection: 'custcollection_my_strings',
    key: 'MY_TITLE'
  })(),
  message: translation.get({
    collection: 'custcollection_my_strings',
    key: 'HELLO_1'
  })({
    params: 'NetSuite'
  }),
  type: message.Type.CONFIRMATION
});
...
// Add additional code
```

translation.get(options)

**Note:** The N/translation module is a beta feature in 2019.1.

**Method Description**

Creates a translator function for a key in the specified Translation Collection and locale.

This method returns a translator function, which is subsequently called with any specified parameters. When you call `translation.get(options)` and do not specify a locale, the method uses the current user's session locale. You can use the `options.locale` parameter to specify another locale. The `translation.Locale` enum lists all locales that are enabled for a company, and you can use these locales in `translation.get(options)`.

**Returns**

`translation.Translator`
**Supported Script Types**
Client and server-side scripts
For additional information, see the help topic SuiteScript 2.0 Script Types.

**Module**
N/translation Module

**Sibling Object Members**
N/translation Module Members

**Since**
2019.1

## Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.collection</td>
<td>string</td>
<td>required</td>
<td>The script ID of the collection.</td>
</tr>
<tr>
<td>options.key</td>
<td>string</td>
<td>required</td>
<td>A valid key from the collection.</td>
</tr>
<tr>
<td>options.locale</td>
<td>string</td>
<td>optional</td>
<td>A valid locale from the translation.Locale enum. If a locale is not specified, the locale from the current session is used as the default locale.</td>
</tr>
</tbody>
</table>

## Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A collection or key parameter is missing.</td>
</tr>
<tr>
<td>INVALID_TRANSLATION_KEY</td>
<td>The format of a specified key is invalid.</td>
</tr>
<tr>
<td>INVALID_TRANSLATION_COLLECTION</td>
<td>The format of a specified collection is invalid.</td>
</tr>
<tr>
<td>INVALID_LOCALE</td>
<td>The format of a specified locale is invalid.</td>
</tr>
<tr>
<td>TRANSLATION_KEY_NOT_FOUND</td>
<td>A specified translation key was not found.</td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/translation Module Script Samples.

```
// Add additional code
...
var myMsg = message.create({
    title: translation.get({
        collection: 'custcollection_my_strings',
        key: 'MY_TITLE'
    })(),
    message: translation.get({
        collection: 'custcollection_my_strings',
        key: 'HELLO_1'
    })({},
        params: 'NetSuite'
    )
});
```
translation.load(options)

**Note:** The N/translation module is a beta feature in 2019.1.

**Method Description**

Creates a `translation.Handle` object with translations for the specified Translation Collections and locales.

This method returns a `translation.Handle` object with translation strings organized by collection and ID. Every node in a `translation.Handle` object is either another `translation.Handle` object or a `translation.Translator` function.

You can load translation strings from multiple Translation Collections in a single call of `translation.load(options)`. You can also specify the keys of individual translation strings that you want to load, instead of loading the entire collection. This method can load a maximum of 1,000 translation strings, regardless of whether the strings are loaded from one collection or multiple collections.

When you load translation strings using `translation.load(options)`, you can specify a list of valid locales for the strings. You can use these locales when you select a locale using `translation.selectLocale(options)`. If you specify more than one locale when you call `translation.load(options)`, the first specified locale in the list is used for the created `translation.Handle` object. If you want to use a different locale from the list, use `translation.selectLocale(options)`, which returns a `translation.Handle` object in the specified locale. You must load a locale using `translation.load(options)` before you can select it using `translation.selectLocale(options)`.

**Returns**

`translation.Handle`

**Supported Script Types**

Client and server-side scripts

For additional information, see the help topic [SuiteScript 2.0 Script Types](#).

**Module**

N/translation Module

**Sibling Object Members**

N/translation Module Members

**Since**

2019.1

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.collections</td>
<td>Object[]</td>
<td>required</td>
<td>A list of <code>translation.Handle</code> objects to load.</td>
</tr>
<tr>
<td>options.collections.alias</td>
<td>string</td>
<td>required</td>
<td>An alias to identify the collection.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>This alias is used by the script to determine the collection to load.</td>
</tr>
<tr>
<td>options.collections.collection</td>
<td>string</td>
<td>required</td>
<td>The script ID of the collection to load.</td>
</tr>
<tr>
<td>options.collections.keys</td>
<td>string[]</td>
<td>optional</td>
<td>A list of translation keys from the collection to load.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Type</td>
<td>Required / Optional</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------</td>
<td>---------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>options.locales</td>
<td>string[]</td>
<td>optional</td>
<td>A list of locales to load the collection in. Use the values in the translation.Locale enum to set this value.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>One of the array parameters (options.collections, options.collections.keys, or options.locales) is not an array.</td>
</tr>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A collection or key parameter is missing.</td>
</tr>
<tr>
<td>INVALID_TRANSLATION_KEY</td>
<td>The format of a specified key is invalid.</td>
</tr>
<tr>
<td>INVALID_TRANSLATION_COLLECTION</td>
<td>The format of a specified collection is invalid.</td>
</tr>
<tr>
<td>INVALID_LOCALE</td>
<td>The format of a specified locale is invalid.</td>
</tr>
<tr>
<td>INVALID_ALIAS</td>
<td>The format of a specified alias is invalid.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/translation Module Script Samples.

```javascript
// Add additional code
...
var localizedStrings = translation.load({
  collections: [{
    alias: 'myCollection',
    collection: 'custcollection_my_strings',
    keys: ['MY_TITLE', 'MY_MESSAGE']
  }
});

var myMsg = message.create({
  title: localizedStrings.myCollection.MY_TITLE(),
  message: localizedStrings.myCollection.MY_MESSAGE(),
  type: message.Type.CONFIRMATION
});

// Add additional code
```

### translation.selectLocale(options)

**Note:** The N/translation module is a beta feature in 2019.1.
This method returns a translation.Handle object that contains the same translation strings as the `options.handle` object, and the strings are in the `options.locale` locale. Before you can use this method to select a locale, the locale must be loaded using the `locales` parameter of `translation.load(options)`.

**Returns**

Translation.Handle

**Supported Script Types**

Client and server-side scripts

For additional information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/translation Module

**Sibling Object Members**

N/translation Module Members

**Since**

2019.1

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>options.handle</code></td>
<td>translation.Handle</td>
<td>required</td>
<td>The translation.Handle object to select a locale for.</td>
</tr>
<tr>
<td><code>options.locale</code></td>
<td>string</td>
<td>required</td>
<td>The locale to select. Use the values in the translation.Locale enum to set this value.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_MISSING_REQD_ARGUMENT</td>
<td>A handle or locale parameter is missing.</td>
</tr>
<tr>
<td>WRONG_PARAMETER_TYPE</td>
<td>The <code>options.handle</code> parameter is not a translation.Handle object.</td>
</tr>
<tr>
<td>INVALID_LOCALE</td>
<td>The specified translation.Handle object uses an unknown or unsupported locale.</td>
</tr>
<tr>
<td>TRANSLATION_HANDLE_IS_IN_AN_ILLEGAL_STATE</td>
<td>The specified translation.Handle object is in an illegal state.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
// Add additional code
...
var germanStrings = translation.load({
    collections: [{
        alias: 'myCollection',
        collection: 'custcollection_my_strings',
        keys: ['MY_TITLE', 'MY_MESSAGE'],
    }],
});
```

---

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/translation Module Script Samples.

var spanishStrings = translation.selectLocale(
    handle: germanStrings,
    locale: translation.Locale.es_ES
));
...  
// Add additional code

## translation.Locale

**Note:** The N/translation module is a beta feature in 2019.1.

| Enum Description | Holds the string values for supported locales for Translation Collections. This enum is used to pass the locale argument to `translation.get(options)` and `translation.selectLocale(options)`. This enum lists all locales that are enabled for a company. This enum also includes two special values: CURRENT and COMPANY_DEFAULT. The CURRENT value represents the current user's locale, and the COMPANY_DEFAULT value represents the default locale for the company. |
|---|

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

| Type | enum |
|---|

| Supported Script Types | Client and server-side scripts For additional information, see the help topic [SuiteScript 2.0 Script Types](https://products.docs.netsuite.com/). |

| Module | N/translation Module |
|---|

| Sibling Object Members | N/translation Module Members |
|---|

| Since | 2019.1 |
|---|

### Values

The following table lists all possible locale values. Typically, only some of these locales will be enabled for a company and available to use with Translation Collections.

| CURRENT | COMPANY_LOCALE | COMPANY_DEFAULT | af_ZA | ar | bg_BG | bn_BD | bs_BA | fr_FR | gu_IN | he_IL | hi_IN | hr_HR | hu_HU | hy_AM | pt_BR | pt_PT | ro_RO | ru_RU | sh_RS | sk_SK | sl_SI |
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/translation Module Script Samples.

```javascript
// Add additional code
...
var myMsg = message.create({
  title: translation.get({
    collection: 'custcollection_my_strings',
    key: 'MY_TITLE',
    locale: translation.Locale.COMPANY_DEFAULT
  })(),
  message: translation.get({
    collection: 'custcollection_my_strings',
    key: 'MY_MESSAGE',
    locale: translation.Locale.COMPANY_DEFAULT
  })(),
  type: message.Type.CONFIRMATION
});
...
// Add additional code
```

**N/ui/dialog Module**

Load the dialog module to create a modal dialog that persists until a button on the dialog is pressed.

**Important:** SuiteScript does not support direct access to the NetSuite UI through the Document Object Model (DOM). The NetSuite UI should only be accessed using SuiteScript APIs.

- **N/ui/dialog Module Members**
N/ui/dialog Module Script Samples

N/ui/dialog Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Property Type / Method Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>dialog.alert(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Creates an Alert dialog with an OK button.</td>
</tr>
<tr>
<td></td>
<td>dialog.###confirm###(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Creates a Confirm dialog with OK and Cancel buttons.</td>
</tr>
<tr>
<td></td>
<td>dialog.create(options)</td>
<td>Promise</td>
<td>Client scripts</td>
<td>Creates a dialog with specified buttons.</td>
</tr>
</tbody>
</table>

N/ui/dialog Module Script Samples

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

Note: To debug client scripts like the following, we recommend that you use Chrome DevTools for Chrome, Firebug debugger for Firefox, or Microsoft Script Debugger for Internet Explorer. For information about these tools, see the documentation provided with each browser.

The following example shows how to create an Alert dialog:

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/ui/dialog'],
function(dialog) {
    var options = {
        title: "I am an Alert",
        message: "Press OK"
    };
    function success(result) {
        console.log("Success with value " + result);
    }
    function failure(reason) {
        console.log("Failure: " + reason);
    }

    dialog.alert(options).then(success).catch(failure);
});
```

The following sample shows how to create a Confirmation dialog:

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/ui/dialog'],
function(dialog) {
    var options = {
        title: "I am a Confirmation",
    };
    function success(result) {
        console.log("This is a Confirmation with value " + result);
    }
    function failure(reason) {
        console.log("Failure: " + reason);
    }

    dialog.confirm(options).then(success).catch(failure);
});
```
message: "Press OK or Cancel"
};
function success(result) {
    console.log("Success with value " + result);
}
function failure(reason) {
    console.log("Failure: " + reason);
}

dialog.confirm(options).then(success).catch(failure);

The following sample shows how to create a dialog with buttons:

/**
 * @NApiVersion 2.x
 */
require(['N/ui/dialog'],
    function(dialog) {
        var button1 = {
            label: 'I am A',
            value: 1
        };
        var button2 = {
            label: 'I am B',
            value: 2
        };
        var button3 = {
            label: 'I am C',
            value: 3
        };
        var options = {
            title: 'Alphabet Test',
            message: 'Which One?',
            buttons: [button1, button2, button3]
        };

        function success(result) {
            console.log("Success with value " + result);
        }
        function failure(reason) {
            console.log("Failure: " + reason);
        }
        dialog.create(options).then(success).catch(failure);
    });

dialog.alert(options)

Method Description
Creates an Alert dialog with an OK button.

Returns
Promise Object. To run a callback function when the OK button is clicked, pass a function to the then portion of the Promise object. When the OK button is clicked, true is passed to the callback. You do not have to utilize the Promise object unless there is an action you want performed after the user clicks the OK button.
Supported Script Types | Client scripts
---|---
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance | None
---|---

Module | N/ui/dialog Module
---|---

Since | 2016.1
---|---

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.title</td>
<td>string</td>
<td>optional</td>
<td>The alert dialog title. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.message</td>
<td>string</td>
<td>optional</td>
<td>The content of the alert dialog. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/dialog Module Script Samples.

```javascript
//Add additional code
...
function success(result) { console.log('Success with value: ' + result) }
function failure(reason) { console.log('Failure: ' + reason) }

dialog.alert({
  title: 'Alert',
  message: 'Click OK to continue.'
}).then(success).catch(failure);
...
//Add additional code
```

dialog.confirm(options)

**Method Description**

Creates a Confirm dialog with OK and Cancel buttons.

**Returns**

Promise Object. To run a callback function when the OK button is pressed, pass a function to the then portion of the Promise object. The value of the pressed button, where OK is true and Cancel is false, is passed to the callback.

**Supported Script Types**

Client scripts

For more information, see the help topic SuiteScript 2.0 Client Script Type.

**Governance**

None

**Module**

N/ui/dialog Module

**Since**

2016.1
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.title</td>
<td>string</td>
<td>optional</td>
<td>The confirmation dialog title. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.message</td>
<td>string</td>
<td>optional</td>
<td>The content of the confirmation dialog. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/dialog Module Script Samples](#).

```javascript
//Add additional code
...
var options = {
  title: 'I am a Confirmation',
  message: 'Press OK or Cancel'
};

function success(result) {
  console.log('Success with value ' + result);
}

function failure(reason) {
  console.log('Failure: ' + reason);
}

dialog.confirm(options).then(success).catch(failure);
...
//Add additional code
```

dialog.create(options)

**Method Description**

Creates a dialog with specified buttons.

**Returns**

`Promise Object`. To run a callback function when a button is pressed, pass a function to the `then` portion of the Promise object. The value of the button pressed is passed to the callback.

**Supported Script Types**

Client scripts

For more information, see the help topic [SuiteScript 2.0 Client Script Type](#).

**Governance**

None

**Module**

N/ui/dialog Module

**Since**

2016.1
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.buttons</td>
<td>string[]</td>
<td>optional</td>
<td>A list of buttons to include in the dialog. Each item in the button list must be a Javascript Object that contains a label and a value property. By default, a single button with the label OK and the value true is used.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.title</td>
<td>string</td>
<td>optional</td>
<td>The dialog title. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.message</td>
<td>string</td>
<td>optional</td>
<td>The content of the dialog. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/dialog Module Script Samples.

```javascript
//Add additional code
...
var options = {
    title: 'Dialog',
    message: 'Click a button to continue.',
    buttons: [
        { label: '1', value: 1 },
        { label: '2', value: 2 },
        { label: '3', value: 3 }
    ]
};

function success(result) { console.log('Success with value: ' + result) }
function failure(reason) { console.log('Failure: ' + reason) }

dialog.create(options).then(success).catch(failure);
...
//Add additional code
```

N/ui/message module

Load the message module to display a message at the top of the screen under the menu bar.

Important: SuiteScript does not support direct access to the NetSuite UI through the Document Object Model (DOM). The NetSuite UI should only be accessed using SuiteScript APIs.

- N/ui/message Members
- Message Object Members
- N/ui/message Module Script Sample
N/ui/message Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>message.Message</td>
<td>void</td>
<td>Client scripts</td>
<td>Encapsulates the Message object that gets created when calling the create method.</td>
</tr>
<tr>
<td>Method</td>
<td>message.create(Message)</td>
<td>Message</td>
<td>Client scripts</td>
<td>Creates a message that can be displayed or hidden near the top of the page.</td>
</tr>
<tr>
<td>Enum</td>
<td>message.Type</td>
<td>enum</td>
<td>Client scripts</td>
<td>Indicates the type of message to display, which specifies the background color of the message and other message indicators.</td>
</tr>
</tbody>
</table>

Message Object Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Message.hide()</td>
<td>void</td>
<td>Client scripts</td>
<td>Hides the message.</td>
</tr>
<tr>
<td></td>
<td>Message.show()</td>
<td>void</td>
<td>Client scripts</td>
<td>Shows the message.</td>
</tr>
</tbody>
</table>

N/ui/message Module Script Sample

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

To debug client scripts like the following, we recommend that you use Chrome DevTools for Chrome, Firebug debugger for Firefox, or Microsoft Script Debugger for Internet Explorer. For information about these tools, see the documentation provided with each browser.

The following example shows how to create a confirmation message:

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/ui/message'],
function(message) {
  var myMsg = message.create({
    title: "My Title",
    message: "My Message",
    type: message.Type.CONFIRMATION
  });

  // will disappear after 5s
  myMsg.show({
    duration: 5000
  });

  var myMsg2 = message.create({
    title: "My Title 2",
    message: "My Message 2",
    type: message.Type.INFORMATION
  });
});
```
```javascript
myMsg2.show();
setTimeout(myMsg2.hide, 15000); // will disappear after 15s

var myMsg3 = message.create({
    title: "My Title 3",
    message: "My Message 3",
    type: message.Type.WARNING
});

myMsg3.show(); // will stay up until hide is called.
```
myMsg.show();
setTimeout(myMsg.hide(), 15000); // hide the message after 15s
...
// Add additional code

### Message.show()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Shows the message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>Client scripts</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/message module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.duration</td>
<td>int</td>
<td>optional</td>
<td>The amount of time, in milliseconds, to show the message. The default is 0, which shows the message until Message.hide() is called. If you specify a duration for message.create() and message.show(), the value from the message.show() method call takes precedence.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/message Module Script Sample.

```javascript
// Add additional code
...
var myMsg = message.create({
    title: "My Title 2",
    message: "My Message 2",
    type: message.Type.INFORMATION
});
myMsg.show({ duration : 1500 });
...
// Add additional code
```

### message.create(options)

| Method Description | Creates a message that can be displayed or hidden near the top of the page. |
Returns  message.Message.

Supported Script Types  Client scripts
For more information, see the help topic SuiteScript 2.0 Client Script Type.

Governance  None

Module  N/ui/message module

Since  2016.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.type</td>
<td>message.Type</td>
<td>required</td>
<td>The message type.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.title</td>
<td>string</td>
<td>optional</td>
<td>The message title. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.message</td>
<td>string</td>
<td>optional</td>
<td>The content of the message. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.duration</td>
<td>int</td>
<td>optional</td>
<td>The amount of time, in milliseconds, to show the message. The default is 0, which shows the message until Message.hide() is called. If you specify a duration for message.create() and message.show(), the value from the message.show() method call takes precedence.</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/message Module Script Sample.

```javascript
//Add additional code
...
var myMsg = message.create({
    title: "My Title",
    message: "My Message",
    type: message.Type.CONFIRMATION
});
...
//Add additional code
```

message.Type

Enum Description  Indicates the type of message to display, which specifies the background color of the message and other message indicators.
Supported Script Types | Client scripts  
| For more information, see the help topic SuiteScript 2.0 Client Script Type.

Module | N/ui/message module

Since | 2016.1

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONFIRMATION</td>
<td>A green background with a checkmark icon.</td>
</tr>
<tr>
<td>INFORMATION</td>
<td>A blue background with an Information icon.</td>
</tr>
<tr>
<td>WARNING</td>
<td>A yellow background with a Warning icon.</td>
</tr>
<tr>
<td>ERROR</td>
<td>A red background with an X icon.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/message Module Script Sample.

```
//Add additional code
...
var myMsg = message.create({
  title: "My Title",
  message: "My Message",
  type: message.Type.CONFIRMATION
});
myMsg.show();
...
//Add additional code
```

N/ui/serverWidget Module

Load the serverWidget module when you want to work with the user interface within NetSuite. You can use Suitelets to build custom pages and wizards that have a NetSuite look-and-feel. You can also create various components of the NetSuite UI (for example, forms, fields, sublists, tabs).

⚠️ **Important:** SuiteScript does not support direct access to the NetSuite UI through the Document Object Model (DOM). The NetSuite UI should only be accessed using SuiteScript APIs.

⚠️ **Important:** When you add a UI object to an existing NetSuite page, to minimize the occurrence of field/object name conflicts, the internal ID that references the object must be prefixed with custpage.

- N/ui/serverWidget Module Members
- Assistant Object Members
- AssistantStep Object Members
N/ui/serverWidget Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>serverWidget.Assistant</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a scriptable, multi-step NetSuite assistant.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##Assistant</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a step within a custom NetSuite assistant.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.Button</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a button that appears in a UI object.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.Field</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a NetSuite field.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##FieldGroup</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a field group.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.Form</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a NetSuite form.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.List</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a list.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##ListColumn</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates list columns.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.Sublist</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates a NetSuite sublist.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.Tab</td>
<td>Object</td>
<td>Suitelets and beforeLoad user events</td>
<td>Encapsulates NetSuite tabs and subtabs.</td>
</tr>
<tr>
<td>Method</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>serverWidget.##createAssistant</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>Creates and returns a new assistant object.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##createForm</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>Creates and returns a new form object.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Enum</td>
<td>serverWidget.##createList</td>
<td>serverWidget.List</td>
<td>Suitelets and beforeLoad user events</td>
<td>Instantiates a List object (specifying the title, and whether to hide the navigation bar)</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##Assistant.submitAction</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for submit actions performed by the user.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##FieldBreakType</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for supported field break types. This enum is used to set the value of the Field.updateBreakType(options) property.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##FieldDisplayType</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for supported field display types. This enum is used to set the value of the Field.updateDisplayType(options) property.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##FieldLayoutType</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for supported field layouts. This enum is used to set the value of the Field.updateLayoutType(options) property.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##FieldType</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the values for supported field types. This enum is used to set the value of the Field.type property.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##FormPageLinkType</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for supported page link types on a form. This enum is used to set the value of the type parameter for Form.addPageLink(options).</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##LayoutJustification</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for supported justification layouts. This enum is used to set the value of the align parameter when List.addColumn(options) is called.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.ListStyle</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for supported list styles. This enum is used to set the value of the List.style property.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##SublistDisplayType</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for supported sublist display types. This enum is used to set the value of the Sublist.displayType property.</td>
</tr>
<tr>
<td></td>
<td>serverWidget.##SublistType</td>
<td>string (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>Holds the string values for valid sublist types. This enum is used to define the type parameter when Form.addSublist(options) is called.</td>
</tr>
</tbody>
</table>

Assistant Object Members

The following members are called on the serverWidget.Assistant object.
<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Property Type / Method</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Assistant.##addField(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a field to an assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##addFieldGroup(options)</td>
<td>serverWidget.##FieldGroup</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a field group to an assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##addStep(options)</td>
<td>serverWidget.##AssistantStep</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a step to an assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##addSublist(options)</td>
<td>serverWidget.Sublist</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a sublist to an assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##addFieldGroupIds()</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets all the field group IDs in an assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.getFieldIds()</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets all the field IDs in an assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##get##Field##Ids##By##Field##Group(fieldGroup)</td>
<td>string[]</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets all field IDs in the assistant field group.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##getLastAction()</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets the last action submitted by the user.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.getLastStep()</td>
<td>serverWidget.##AssistantStep</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets the step that the last submitted action came from.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.getNextStep()</td>
<td>serverWidget.##AssistantStep</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets the next step prompted by the assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##getStep(options)</td>
<td>serverWidget.##AssistantStep</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns a step in an assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##getStepCount()</td>
<td>serverWidget.##AssistantStep</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets the total count of steps in the assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.getSteps()</td>
<td>serverWidget.##AssistantStep[]</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets all the steps in the assistant.</td>
</tr>
<tr>
<td>Method</td>
<td>Assistant.##getSublist(options)</td>
<td>serverWidget.Sublist</td>
<td>Suitelets and beforeLoad user events</td>
<td>Get a Sublist object from its ID.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Property Type / Method Return Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-----------------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Assistant</td>
<td>getSublistIds()</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets all the sublist IDs in an assistant.</td>
</tr>
<tr>
<td>Assistant</td>
<td>hasErrorHtml()</td>
<td>boolean</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the assistant threw an error.</td>
</tr>
<tr>
<td>Assistant</td>
<td>isFinished()</td>
<td>boolean</td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the status of the assistant. If set to true, the assistant is finished.</td>
</tr>
<tr>
<td>Assistant</td>
<td>#sendRedirect(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Manages redirects in an assistant.</td>
</tr>
<tr>
<td>Assistant</td>
<td>#setSplash(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Define a splash message.</td>
</tr>
<tr>
<td>Assistant</td>
<td>#updateDefaultValues(values)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the default values of an array of fields that are specific to the assistant.</td>
</tr>
<tr>
<td>Assistant</td>
<td>clientScriptFileId</td>
<td>number</td>
<td>Suitelets and beforeLoad user events</td>
<td>The file cabinet ID of client script file to be used in this assistant.</td>
</tr>
<tr>
<td>Assistant</td>
<td>clientScriptModulePath</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The relative path to the client script file to be used in this assistant.</td>
</tr>
<tr>
<td>Assistant</td>
<td>currentStep</td>
<td>serverWidgetAssistantStep</td>
<td>Suitelets and beforeLoad user events</td>
<td>Identifies the current step.</td>
</tr>
<tr>
<td>Assistant</td>
<td>errorHtml</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The error message text.</td>
</tr>
<tr>
<td>Assistant</td>
<td>finishedHtml</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The text displayed after an assistant is finished.</td>
</tr>
<tr>
<td>Assistant</td>
<td>hideAddToShortcutsLink</td>
<td>boolean</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the Add to Shortcuts Link is displayed in the UI.</td>
</tr>
<tr>
<td>Assistant</td>
<td>hideStepNumber</td>
<td>boolean</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the current and total step numbers are displayed in the UI.</td>
</tr>
<tr>
<td>Assistant</td>
<td>isNotOrdered</td>
<td>boolean</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether assistant steps are ordered or unordered.</td>
</tr>
<tr>
<td>Assistant</td>
<td>title</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The title of an assistant.</td>
</tr>
</tbody>
</table>

AssistantStep Object Members

The following members are called on the `serverWidget.AssistantStep` object.
## N/ui/serverWidget Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>AssistantStep.getFieldIds()</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets all the field IDs in an assistant step.</td>
</tr>
<tr>
<td>Method</td>
<td>AssistantStep.getLineCount(options)</td>
<td>number</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets the number of lines previously entered by a user in a step.</td>
</tr>
<tr>
<td>Method</td>
<td>AssistantStep.getLineCount()</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets all the field IDs in a list.</td>
</tr>
<tr>
<td>Method</td>
<td>AssistantStep.getSubmittedSublistIds()</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets the IDs for all the sublist fields (line items) in a step.</td>
</tr>
<tr>
<td>Method</td>
<td>AssistantStep.getValue(options)</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets the current value of a field in a step.</td>
</tr>
<tr>
<td>Property</td>
<td>AssistantStep.helpText</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The help text for a step.</td>
</tr>
<tr>
<td>Property</td>
<td>AssistantStep.id</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The internal ID of the step.</td>
</tr>
<tr>
<td>Property</td>
<td>AssistantStep.label</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The label for a step.</td>
</tr>
<tr>
<td>Property</td>
<td>AssistantStep.stepNumber</td>
<td>number</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates where this step appears sequentially in an assistant.</td>
</tr>
</tbody>
</table>

### Button Object Members

The following members are called on the `serverWidget.Button` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Property Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Button.isDisabled</td>
<td>boolean true</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether a button is grayed-out and disabled.</td>
</tr>
<tr>
<td></td>
<td>Button.isHidden</td>
<td>boolean true</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the button is hidden in the UI.</td>
</tr>
<tr>
<td></td>
<td>Button.label</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The label for the button.</td>
</tr>
</tbody>
</table>
Field Object Members

The following members are called on the `serverWidget.Field` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Field.addSelectOption(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a select option to a dropdown list for a selectable field.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.getSelectOptions(options)</td>
<td>object[]</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns the internal ID and label of the options for a select field as name/value pairs.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.setHelpText(options)</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the help text that appears in the field help popup.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.updateBreakType(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Updates the break type used to add a break in flow layout for the field.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.updateDisplaySize(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Updates the height and width for the field.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.updateDisplayType(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Updates the type of display for the field.</td>
</tr>
<tr>
<td>Method</td>
<td>Field.updateLayoutType(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Updates the layout type for the field.</td>
</tr>
<tr>
<td>Property</td>
<td>Field.alias</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The alias used to set the field value.</td>
</tr>
<tr>
<td>Property</td>
<td>Field.defaultValue</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The default value for the field.</td>
</tr>
<tr>
<td>Property</td>
<td>Field.id</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The internal ID for the field.</td>
</tr>
<tr>
<td>Property</td>
<td>Field.isMandatory</td>
<td>boolean true</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the field is mandatory.</td>
</tr>
<tr>
<td>Property</td>
<td>Field.label</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets or sets the label for the field.</td>
</tr>
<tr>
<td>Property</td>
<td>Field.linkText</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The text displayed for a link in place of the URL.</td>
</tr>
<tr>
<td>Property</td>
<td>FieldGroup.isBorderHidden</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>The maximum length, in characters, for the field.</td>
</tr>
</tbody>
</table>
### N/ui/serverWidget Module

#### Field Object Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field.padding</td>
<td>number</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>The number of empty vertical character spaces above the field.</td>
</tr>
<tr>
<td>Field.richTextHeight</td>
<td>number</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>The height of a rich text field, in pixels.</td>
</tr>
<tr>
<td>Field.richTextWidth</td>
<td>number</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>The width of a rich text field, in pixels.</td>
</tr>
<tr>
<td>Field.type</td>
<td>string</td>
<td></td>
<td>Suitelets and beforeLoad user events</td>
<td>The type of field.</td>
</tr>
</tbody>
</table>

#### FieldGroup Object Members

The following members are called on the serverWidget.FieldGroup object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Property Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>FieldGroup.###isBorderHidden</td>
<td>boolean true</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether a border appears around the field group.</td>
</tr>
<tr>
<td></td>
<td>FieldGroup.isCollapsible</td>
<td>boolean true</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the field group is collapsible.</td>
</tr>
<tr>
<td></td>
<td>FieldGroup.isCollapsed</td>
<td>boolean true</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the field group is initially collapsed or expanded in the default view.</td>
</tr>
<tr>
<td></td>
<td>FieldGroup.###isSingleColumn</td>
<td>boolean true</td>
<td>Suitelets and beforeLoad user events</td>
<td>Indicates whether the field group is aligned.</td>
</tr>
<tr>
<td></td>
<td>FieldGroup.label</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The label for the field group.</td>
</tr>
</tbody>
</table>

#### Form Object Members

The following members are called on the serverWidget.Form object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Property Type / Method Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Form.###addButton(options)</td>
<td>serverWidget.Button</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a button to the form.</td>
</tr>
<tr>
<td></td>
<td>Form.###addCredentialField(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a field that store credentials in NetSuite for invoking services provided by third parties.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Property Type / Method Return Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-----------------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Form.<strong>addField</strong>(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a field to the form.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addFieldGroup</strong>(options)</td>
<td>serverWidget.FieldGroup</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a group of fields to the form.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addPageMessage</strong>(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Shows a message on a form in view mode. You can use this method to show a message on a form based on its user event script context.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addPageLink</strong>(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a link to a form.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addResetButton</strong>(options)</td>
<td>serverWidget.Button</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a reset button to a form that clears user input.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addSecretKeyField</strong>(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Add a secret key field to the form.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addSublist</strong>(options)</td>
<td>serverWidget.Sublist</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a sublist to the form.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addSubmitButton</strong>(options)</td>
<td>serverWidget.Button</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a submit button to a form that saves user inputs.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>addSubtab</strong>(options)</td>
<td>serverWidget.Tab</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a subtab to a form.</td>
<td></td>
</tr>
<tr>
<td>Form.addTab(options)</td>
<td>serverWidget.Tab</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a tab to a form.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>getButton</strong>(options)</td>
<td>serverWidget.Button</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns a button by internal ID.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>getField</strong>(options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns a field by internal ID.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>getSublist</strong>(options)</td>
<td>serverWidget.Sublist</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns a sublist by internal ID.</td>
<td></td>
</tr>
<tr>
<td>Form.<strong>getSubtab</strong>(options)</td>
<td>serverWidget.Tab</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns a subtab by internal ID.</td>
<td></td>
</tr>
<tr>
<td>Form.getTab(options)</td>
<td>serverWidget.Tab</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns a tab object from its internal ID.</td>
<td></td>
</tr>
<tr>
<td>Form.getTabs()</td>
<td>serverWidget.Tab[]</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns an array of all the tabs in a form.</td>
<td></td>
</tr>
</tbody>
</table>
### N/ui/serverWidget Module

#### SuiteScript 2.0 API Reference

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Property Type / Method Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form.###insertField###(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Inserts a field before another field within a form.</td>
<td></td>
</tr>
<tr>
<td>Form.###insertSublist###(options)</td>
<td>serverWidget.Sublist</td>
<td>Suitelets and beforeLoad user events</td>
<td>Inserts a sublist before another sublist on a form.</td>
<td></td>
</tr>
<tr>
<td>Form.###insertSubtab###(options)</td>
<td>serverWidget.Tab</td>
<td>Suitelets and beforeLoad user events</td>
<td>Inserts a subtab before another subtab on a form.</td>
<td></td>
</tr>
<tr>
<td>Form.###insertTab###(options)</td>
<td>serverWidget.Tab</td>
<td>Suitelets and beforeLoad user events</td>
<td>Inserts a tab before another tab on a form.</td>
<td></td>
</tr>
<tr>
<td>Form.###removeButton###(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Removes a button from a form.</td>
<td></td>
</tr>
<tr>
<td>Form.###update###Default###Values###(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the default values of many fields on a form.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property</th>
<th>Name</th>
<th>Property Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form.clientScriptField</td>
<td>number</td>
<td>Suitelets and beforeLoad user events</td>
<td>The file cabinet ID of client script file to be used in this form.</td>
<td></td>
</tr>
<tr>
<td>Form.###client###Script###Module###Path ###(options)</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The relative path to the client script file to be used in this form.</td>
<td></td>
</tr>
<tr>
<td>Form.title</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The title used for the form.</td>
<td></td>
</tr>
</tbody>
</table>

#### List Object Members

The following members are called on the serverWidget.List object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Property Type / Method Return Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>List.###addButton###(options)</td>
<td>serverWidget.Button</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a button to a list.</td>
</tr>
<tr>
<td>List.###addColumn###(options)</td>
<td>serverWidget.ListColumn</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a column to a list.</td>
<td></td>
</tr>
<tr>
<td>List.###addEditColumn###(options)</td>
<td>serverWidget.ListColumn</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a column containing Edit or Edit/View links to a Suitelet or Portlet list.</td>
<td></td>
</tr>
<tr>
<td>List.###addPageLink###(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a link to a list.</td>
<td></td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Property Type / Method Return Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------</td>
<td>------------------------------------</td>
<td>-------------------------------------------------</td>
<td>-------------------------------------------------</td>
</tr>
<tr>
<td>Property</td>
<td>List.addRow(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a single row to a list.</td>
</tr>
<tr>
<td></td>
<td>List.addRows(options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds multiple rows to a list.</td>
</tr>
<tr>
<td></td>
<td>List.clientScriptFileId</td>
<td>number</td>
<td>Suitelets and beforeLoad user events</td>
<td>The file cabinet ID of client script file to be used in this list.</td>
</tr>
<tr>
<td></td>
<td>List.<strong>client</strong>.<strong>Script</strong>.<strong>Module</strong>.<strong>Path</strong></td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The relative path to the client script file to be used in this list.</td>
</tr>
<tr>
<td></td>
<td>List.style</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the display style for this list.</td>
</tr>
<tr>
<td></td>
<td>List.title</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the List title.</td>
</tr>
</tbody>
</table>

**ListColumn Object Members**

The following members are called on the `serverWidget.ListColumn` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>ListColumn.<strong>addParamToURL</strong>(options)</td>
<td>serverWidget.<strong>ListColumn</strong></td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a URL parameter (optionally defined per row) to the list column's URL.</td>
</tr>
<tr>
<td></td>
<td>ListColumn.<strong>setURL</strong>(options)</td>
<td>serverWidget.<strong>ListColumn</strong></td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the base URL for the list column.</td>
</tr>
<tr>
<td>Property</td>
<td>ListColumn.label</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The label of this list column.</td>
</tr>
</tbody>
</table>

**Sublist Object Members**

The following members are called on the `serverWidget.Sublist` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Sublist.<strong>addButton</strong>(options)</td>
<td>serverWidget.<strong>Button</strong></td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a button to a sublist.</td>
</tr>
<tr>
<td></td>
<td>Sublist.<strong>addField</strong>(options)</td>
<td>serverWidget.<strong>Field</strong></td>
<td>Suitelets and beforeLoad user events</td>
<td>Add a field to a sublist.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------</td>
<td>--------------------------</td>
<td>-------------------------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Sublist</td>
<td><strong>add</strong> Mark All Button</td>
<td>serverWidget.Button</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a Mark All or Unmark All button.</td>
</tr>
<tr>
<td>Sublist</td>
<td><strong>add</strong> Refresh Button</td>
<td>serverWidget.Button</td>
<td>Suitelets and beforeLoad user events</td>
<td>Adds a Reset button.</td>
</tr>
<tr>
<td>Sublist</td>
<td><strong>getField</strong> (options)</td>
<td>serverWidget.Field</td>
<td>Suitelets and beforeLoad user events</td>
<td>Returns a Field object on a specified sublist.</td>
</tr>
<tr>
<td>Sublist</td>
<td><strong>getSublistValue</strong> (options)</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>Gets a field value on a sublist.</td>
</tr>
<tr>
<td>Sublist</td>
<td><strong>setSublistValue</strong> (options)</td>
<td>void</td>
<td>Suitelets and beforeLoad user events</td>
<td>Sets the value of a sublist field.</td>
</tr>
<tr>
<td>Sublist</td>
<td><strong>update</strong> Totalling Field ID (options)</td>
<td>serverWidget.Sublist</td>
<td>Suitelets and beforeLoad user events</td>
<td>Updates the ID of a field designated as a totalling column, which is used to calculate and display a running total for the sublist.</td>
</tr>
<tr>
<td>Sublist</td>
<td><strong>update</strong> Unique Field ID (options)</td>
<td>serverWidget.Sublist</td>
<td>Suitelets and beforeLoad user events</td>
<td>Updates a field ID that is to have unique values across the rows in the sublist.</td>
</tr>
<tr>
<td>Sublist.displayType</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The display style for a sublist.</td>
<td></td>
</tr>
<tr>
<td>Sublist.helpText</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The inline help text for a sublist.</td>
<td></td>
</tr>
<tr>
<td>Sublist.label</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The label for a sublist.</td>
<td></td>
</tr>
<tr>
<td>Sublist.lineCount</td>
<td>number (read-only)</td>
<td>Suitelets and beforeLoad user events</td>
<td>The number of line items in a sublist.</td>
<td></td>
</tr>
</tbody>
</table>

### Tab Object Members

The following members are called on the serverWidget.Tab object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Tab.helpText</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The inline help text for a tab or subtab.</td>
</tr>
<tr>
<td></td>
<td>Tab.label</td>
<td>string</td>
<td>Suitelets and beforeLoad user events</td>
<td>The label for a tab or subtab.</td>
</tr>
</tbody>
</table>

### N/ui/serverWidget Module Script Samples

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.
Example 1
The following code creates a Suitelet that generates a sample form with a submit button, fields, and an inline editor sublist:

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
define(["N/ui/serverWidget"],
    function(serverWidget) {
        function onRequest(context) {
            if (context.request.method === 'GET') {
                var form = serverWidget.createForm({
                    title: 'Simple Form'
                });
                var field = form.addField({
                    id: 'custpage_text',
                    type: serverWidget.FieldType.TEXT,
                    label: 'Text'
                });
                field.layoutType = serverWidget.FieldLayoutType.NORMAL;
                field.updateBreakType({breakType: serverWidget.FieldBreakType.STARTCOL});
                form.addField({
                    id: 'custpage_date',
                    type: serverWidget.FieldType.DATE,
                    label: 'Date'
                });
                form.addField({
                    id: 'custpage_currencyfield',
                    type: serverWidget.FieldType.CURRENCY,
                    label: 'Currency'
                });
                var select = form.addField({
                    id: 'custpage_selectfield',
                    type: serverWidget.FieldType.SELECT,
                    label: 'Select'
                });
                select.addSelectOption({
                    value: 'a',
                    text: 'Albert'
                });
                select.addSelectOption({
                    value: 'b',
                    text: 'Baron'
                });
                var sublist = form.addSublist({
                    id: 'sublist',
                    type: serverWidget.SublistType.INLINEEDITOR,
                    label: 'Inline Editor Sublist'
                });
                sublist.addField({
                    id: 'sublist1',
                    type: serverWidget.FieldType.DATE,
                    label: 'Date'
                });
            }
        }
    });
```
N/ui/serverWidget Module

sublist.addField({
id: 'sublist2',
type: serverWidget.FieldType.TEXT,
label: 'Text'
});
form.addSubmitButton({
label: 'Submit Button'
});
context.response.writePage(form);
} else {
var delimiter = /\u0001/;
var textField = context.request.parameters.textfield;
var dateField = context.request.parameters.datefield;
var currencyField = context.request.parameters.currencyfield;
var selectField = context.request.parameters.selectfield;
var sublistData = context.request.parameters.sublistdata.split(delimiter);
var sublistField1 = sublistData[0];
var sublistField2 = sublistData[1];
context.response.write('You have entered: ' + textField + ' ' + dateField + ' '
+ currencyField + ' ' + selectField + ' ' + sublistField1 + ' ' + sublistField2);
}
}
return {
onRequest: onRequest
};
});

Example 2
The following code creates Suitelet that generates a customer survey form with inline HTML fields,
radio fields, a submit button, and a reset button.
/**
* @NApiVersion 2.0
* @NScriptType suitelet
*/
define(['N/ui/serverWidget'], function(serverWidget){
function onRequest(context){
var form = serverWidget.createForm({
title: 'Thank you for your interest in Wolfe Electronics',
hideNavBar: true
});
var htmlHeader = form.addField({
id: 'custpage_header',
type: serverWidget.FieldType.INLINEHTML,
label: ' '
}).updateLayoutType({
layoutType: serverWidget.FieldLayoutType.OUTSIDEABOVE
}).updateBreakType({
breakType: serverWidget.FieldBreakType.STARTROW
}).defaultValue = "<p style='font-size:20px'>We pride ourselves on providing the best" +
"services and customer satisfaction. Please take a moment to fill out our survey.</p><br><br>";

SuiteScript 2.0 API Reference

962


var htmlInstruct = form.addField({
    id: 'custpage_p1',
    type: serverWidget.FieldType.INLINEHTML,
    label: ' ' 
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.OUTSIDEABOVE
}).updateBreakType({
    breakType: serverWidget.FieldBreakType.STARTROW
}).defaultValue = '<p style='font-size:14px'>When answering questions on a scale of 1 to 5, "1 = Greatly Unsatisfied and 5 = Greatly Satisfied."</p>';

form.addField({
    id: 'custpageLblProductrating',
    type: serverWidget.FieldType.INLINEHTML,
    label: ' ' 
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.NORMAL
}).updateBreakType({
    breakType: serverWidget.FieldBreakType.STARTROW
}).defaultValue = '<p style='font-size:14px'>How would you rate your satisfaction with our products?</p>';

form.addField({
    id: 'custpageRdoproductrating',
    type: serverWidget.FieldType.RADIO,
    label: '1',
    source: 'p1'
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.STARTROW
});

form.addField({
    id: 'custpageRdoproductrating',
    type: serverWidget.FieldType.RADIO,
    label: '2',
    source: 'p2'
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.MIDROW
});

form.addField({
    id: 'custpageRdoproductrating',
    type: serverWidget.FieldType.RADIO,
    label: '3',
    source: 'p3'
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.MIDROW
});

form.addField({
    id: 'custpageRdoproductrating',
    type: serverWidget.FieldType.RADIO,
    label: '4',
    source: 'p4'
}).updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.MIDROW
});
type: serverWidget.FieldType.RADIO,
label: '5',
source: 'p5'
}).updateLayoutType(
    layoutType: serverWidget.FieldLayoutType.ENDROW
));
form.addField(
    id: 'custpage_lblservicerating',
    type: serverWidget.FieldType.INLINEHTML,
    label: '
').updateLayoutType(
    layoutType: serverWidget.FieldLayoutType.NORMAL
)).updateBreakType(
    breakType: serverWidget.FieldBreakType.STARTROW
)).defaultValue = '<p style='font-size:14px'>How would you rate your satisfaction with our services?</p>);
form.addField(
    id: 'custpage_rdoservicerating',
    type: serverWidget.FieldType.RADIO,
    label: '1',
    source: 'p1'
}).updateLayoutType(
    layoutType: serverWidget.FieldLayoutType.STARTROW
));
form.addField(
    id: 'custpage_rdoservicerating',
    type: serverWidget.FieldType.RADIO,
    label: '2',
    source: 'p2'
}).updateLayoutType(
    layoutType: serverWidget.FieldLayoutType.MIDROW
));
form.addField(
    id: 'custpage_rdoservicerating',
    type: serverWidget.FieldType.RADIO,
    label: '3',
    source: 'p3'
}).updateLayoutType(
    layoutType: serverWidget.FieldLayoutType.MIDROW
));
form.addField(
    id: 'custpage_rdoservicerating',
    type: serverWidget.FieldType.RADIO,
    label: '4',
    source: 'p4'
}).updateLayoutType(
    layoutType: serverWidget.FieldLayoutType.MIDROW
));
form.addField(
    id: 'custpage_rdoservicerating',
    type: serverWidget.FieldType.RADIO,
    label: '5',
    source: 'p5'
}).updateLayoutType(
    layoutType: serverWidget.FieldLayoutType.ENDROW
);
serverWidget.Assistant

Object Description
Encapsulates a scriptable, multi-step NetSuite assistant. An assistant contains a series of step that a user must complete to accomplish a larger goal. An assistant can be sequential, or non-sequential and include optional steps. Each page of the assistant is defined by a step. All data and states for an assistant are tracked automatically throughout the user's session until completion of the assistant.

You can create a new assistant with the serverWidget.createAssistant(options) method. After you create an Assistant object, you can:

- Build and run an assistant in your NetSuite account.
- Add a variety of scriptable elements to the assistant including fields, steps, buttons, tabs, and sublists.

For a complete list of this object's methods and properties, see Assistant Object Members.

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Module
N/ui/serverWidget Module

Since
2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant'
});
...
//Add additional code
```
## Assistant.addField(options)

**Method Description**
Adds a field to an assistant. Use fields to record or display information specific to your needs.

**Returns**
serverWidget.Field object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

### 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>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID for this field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The field type. Use the serverWidget.FieldType enum to set this value.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.source</td>
<td>string</td>
<td>optional</td>
<td>The internalId or scriptId of the source list for this field. Use the serverWidget.FieldType enum to set this value.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.

**Note:** If you have set the type parameter to `SELECT`, and you want to add custom options to the select field, you must set source to NULL. Then, when a value is specified, the value will populate the options from the source.

**Important:** Long text fields created with SuiteScript have a character limit of 100,000. Long text fields created with Suitebuilder have a character limit of 1,000,000.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.container</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of the field group to place this field in.</td>
</tr>
</tbody>
</table>

**Note:** If you want to add custom options on a select field, you must set the source parameter to NULL.

**Important:** After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with Field.addSelectOption(options). The select values are determined by the source record or list.

**Syntax**

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant'
});
assistant.addField({
  id : 'idname',
  type : serverWidget.FieldType.TEXT,
  label : 'Sample label'
});
...
//Add additional code
```

**Assistant.addFieldGroup(options)**

**Method Description**

Adds a field group to the assistant. A field group is a collection of fields that can be displayed in a one or two column format. Assign a field to a field group in order to label, hide or collapse a group of fields.

By default, the field group is collapsible and appears expanded on the assistant page. To change this behavior, set the FieldGroup.isCollapsed and FieldGroup.isCollapsible properties.

**Returns**

serverWidget.FieldGroup object

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID for the field group.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for the field group.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addFieldGroup({
    id : 'idname',
    label : 'Sample label'
});
...
//Add additional code
```

Assistant.addStep(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Adds a step to an assistant. Steps define each page of the assistant. Use Assistant.isNotOrdered to control if the steps must be completed sequentially or in no specific order. If you want to create help text for the step, you can use AssistantStep.helpText on the object returned.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>serverWidget.AssistantStep object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID for this step (for example, 'entercontacts').</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this step (for example, 'Enter Contacts'). By default, the step appears vertically in the left panel of the assistant.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
...
//Add additional code
```

**Assistant.addSublist(options)**

**Method Description**

Adds a sublist to an assistant.

**Note:** Only inline editor sublists are added. Other sublist types are not supported.

**Returns**

*serverWidget.Sublist object*

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

*N/ui/serverWidget Module*

**Since**

2015.2
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID for the sublist.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for the sublist.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The type of sublist to add. Currently, only the sublist type of INLINEEDITOR can be added. For more information about this type of sublist, see serverWidget.SublistType.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addSublist({
    id : 'idname',
    label : 'Sample label',
    type : serverWidget.SublistType.INLINEEDITOR
});
...
//Add additional code
```

Assistant.getField(options)

**Method Description**

Returns a field object on an assistant page.

**Returns**

serverWidget.Field

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2
### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant',
});
assistant.addField({
  id : 'idname',
  type : serverWidget.FieldType.TEXT,
  label : 'Sample label',
});
var field = assistant.getField({
  id: 'idname' ...
});
//Add additional code
```

### Assistant.getFieldGroup(options)

**Method Description**

Returns a field group on an assistant page.

**Returns**

`serverWidget.FieldGroup` object

**Supported Script Types**

`SuiteScript 2.0 Suitelet Script Type` and `SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))`

**Governance**

None

**Module**

`N/ui/serverWidget Module`

**Since**

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the field group.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Assistant.getFieldGroupIds()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Retrieves all the internal IDs for field groups in an assistant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string[]</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title: 'Simple Assistant'
});
assistant.addFieldGroup({
  id: 'idname',
  label: 'Sample label'
});
var fieldgroup = assistant.getFieldGroup({
  id: 'idname'
});
...
//Add additional code
```
Assistant.getFieldIds()

**Method Description**

Gets all the internal IDs for fields in an assistant.

**Returns**

string[]

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant(
    title : 'Simple Assistant'
);
assistant.addField(
    id : 'idname',
    label : 'Sample label'
);
var fieldid = assistant.getFieldIds();
...
//Add additional code
```

Assistant.getFieldIdsByFieldGroup(fieldGroup)

**Method Description**

Gets all field IDs in the assistant field group.

**Returns**

string[]

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2016.1

**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>fieldGroup</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the field group.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>
## Assistant.getLastAction()

**Method Description**

Gets the last action taken by the user. To identify the step that the last action came from, use `Assistant getLastStep()`.

**Returns**

string

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2
Assistant.getLastStep()

**Method Description**
Gets the step that the last submitted action came from.

**Returns**
A `serverWidget.AssistantStep` object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

**Syntax**

＞ Insurance 

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```
//Add additional code
... 
var assistant = serverWidget.createAssistant({
    title: 'Simple Assistant'
});
... 
var lastStep = assistant.getLastStep();
... 
//Add additional code
```

Assistant.getNextStep()

**Method Description**
Gets the next step corresponding to the user’s last submitted action in the assistant.

If you need information about the last step, use `Assistant.getLastStep()` before you use this method.

**Returns**
`serverWidget.AssistantStep` object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2
Assistant.getStep(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>serverWidget.AssistantStep object</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
<td>None</td>
<td>N/ui/serverWidget Module</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the step.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
var nextStep = assistant.getNextStep();
...
//Add additional code
```
### Assistant.getStepCount()

**Method Description**
Gets the total number of steps in an assistant.

<table>
<thead>
<tr>
<th>Returns</th>
<th>The total count of assistant steps as a number</th>
</tr>
</thead>
</table>

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var numSteps = assistant.getStepCount();
...  
//Add additional code
```

### Assistant.getSteps()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets all the steps in an assistant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>serverWidget.AssistantStep[] object</td>
</tr>
</tbody>
</table>

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module
## Assistant.getSublist(options)

### Method Description

Returns a sublist in an assistant.

### Returns

serverWidget.Sublist object

### Supported Script Types

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type
(beforeLoad(scriptContext))

### Governance

None

### Module

N/ui/serverWidget Module

### Since

2015.2

### 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>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant(
    title: 'Simple Assistant'
);
assistant.addStep(
    id: 'idname',
    label: 'Sample label'
);
var steps = assistant.getSteps();
...
//Add additional code
```
var assistant = serverWidget.createAssistant({
  title: 'Simple Assistant'
});
assistant.addSublist({
  id: 'idname',
  label: 'Sample label',
  type: serverWidget.SublistType.LIST
});
var sublist = assistant.getSublist({
  id: 'idname'
});
...
//Add additional code

Assistant.getSublistIds()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the IDs for all the sublists in an assistant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string[]</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title: 'Simple Assistant'
});
assistant.addSublist({
  id: 'idname',
  label: 'Sample label',
  type: serverWidget.SublistType.LIST
});
var sublistid = assistant.getSublistIds();
...
//Add additional code
```

Assistant.hasErrorHtml()

| Method Description | Determine whether an assistant has an error message to display for the current step. |
### Assistant.errorHtml()

**Returns**

boolean true if Assistant.errorHtml contains a value.

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title: 'Simple Assistant'
});
...
if (assistant.hasErrorHtml()) {
  ...
}
...
//Add additional code
```

### Assistant.isFinished()

**Method Description**

Indicates whether all steps in an assistant are completed.

If set to true, the assistant is finished and a completion message displays. To set the text for the completion message, use the Assistant.finishedHtml property.

**Returns**

boolean true | false

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
```
Assistant.sendRedirect(options)

**Method Description**  
Manages redirects in an assistant. This method also addresses the case in which one assistant redirects to another assistant. In this scenario, the second assistant must return to the first assistant if the user Cancels or Finishes. This method, when used in the second assistant, ensures that users are redirected back to the first assistant.

**Returns**  
void

**Supported Script Types**  
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**  
None

**Module**  
N/ui/serverWidget Module

**Since**  
2015.2

### 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>options.response</td>
<td>response</td>
<td>required</td>
<td>The response that redirects the user.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
var assistant = serverWidget.createAssistant({
    title: 'Simple Assistant'
});
...
if (assistant.isFinished()) {
    ...
    ...
}

//Add additional code
```

```javascript
Assistant.sendRedirect(options)
```

```javascript
//Add additional code
```

```javascript
var assistant = serverWidget.createAssistant({
    title: 'Small Business Setup Assistant',
    hideNavBar: true
});
```
if (request.method === 'POST') {
    if (assistant.getLastAction() === 'finish') {
        assistant.finishedHtml = 'Completed!';
        assistant.sendRedirect(
            response: response
        );
    }
}
...
//Add additional code

Assistant.setSplash(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Defines a splash message.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.title</td>
<td>string</td>
<td>required</td>
<td>The title of the splash screen.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.text1</td>
<td>string</td>
<td>required</td>
<td>Text for the splash screen</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.text2</td>
<td>string</td>
<td>optional</td>
<td>Text for a second column on the splash screen, if desired.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.setSplash({
    title: 'Welcome Title!,'
    text1: 'An explanation of what this assistant accomplishes.,'
    text2: 'Some parting words.'
});
```
Assistant.updateDefaultValues(values)

Method Description | Sets the default values of an array of fields that are specific to the assistant.
--- | ---
Returns | void
Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
Governance | None
Module | N/ui/serverWidget Module
Since | 2016.1

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>values</td>
<td>object[]</td>
<td>required</td>
<td>An array of fields to update.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant(
  title : 'Simple Assistant'
));
assistant.addField(
  id : 'idname',
  type : serverWidget.FieldType.TEXT,
  label : 'Sample label'
));
assistant.updateDefaultValues(
  idname : 'New Default Value'
));
...
//Add additional code
```

Assistant.clientScriptFileId

Property Description | The file cabinet ID of client script file to be used in this assistant.
--- | ---

**Type**

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY_VALUE_CONFLICT</td>
<td>You attempted to set this value when the Assistant.clientScriptModulePath property value has already been specified. For more information, see Assistant.clientScriptModulePath.</td>
</tr>
</tbody>
</table>

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant'
});
assistant.clientScriptId = 32;
...
//Add additional code
```

**Assistant.clientScriptModulePath**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The relative path to the client script file to be used in this assistant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY_VALUE_CONFLICT</td>
<td>You attempted to set this value when the Assistant.clientScriptFileId property value has already been specified. For more information, see Assistant.clientScriptFileId.</td>
</tr>
</tbody>
</table>
Assistant.currentStep

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identifies the current step. You can set any step as the current step.</td>
</tr>
<tr>
<td>Type</td>
<td>serverWidget.AssistantStep (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Assistant.errorHtml

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Error message text for the current step. Optionally, you can use HTML tags to format the message.</td>
</tr>
</tbody>
</table>
Assistant.finishedHtml

**Property Description**: The text to display after the assistant finishes. For example “You have completed the Small Business Setup Assistant. Take the rest of the day off”.

To trigger display of the completion message, call Assistant.isFinished().

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
assistant.errorHtml = "You have <b>not</b> filled out the required fields. Please go back.;
...
//Add additional code
```
Assistant.hideAddToShortcutsLink

Property Description: Indicates whether to show or hide the Add to Shortcuts link that appears in the top-right corner of an assistant page. By default, the value is false, which means the Add to Shortcuts link is visible in the UI. If set to true, the Add To Shortcuts link is not visible on an Assistant page.

Type: boolean true | false

Supported Script Types: SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Module: N/ui/serverWidget Module

Since: 2015.2

Syntax

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title: 'Simple Assistant'
});
assistant.hideAddToShortcutsLink = true;
...
//Add additional code
```

Assistant.hideStepNumber

Property Description: Indicates whether assistant steps are displayed with numbers. By default, the value is false, which means that steps are numbered. If set to true, the assistant does not use step numbers. To change step ordering, set Assistant.isNotOrdered.

Type: boolean true | false

Supported Script Types: SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Module: N/ui/serverWidget Module

Since: 2015.2
Assistant.isNotOrdered

**Property Description**
Indicates whether steps must be completed in a particular sequence. If steps are ordered, users must complete the current step before proceeding to the next step. The default value is `false`, which means the steps are ordered. Ordered steps appear vertically in the left panel of the assistant. If set to `true`, steps can be completed in any order. In the UI, unordered steps appear horizontally and below the assistant title.

**Type**
`boolean` (`true` | `false`)

**Supported Script Types**
`SuiteScript 2.0 Suitelet Script Type` and `SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))`

**Module**
`N/ui/serverWidget Module`

**Since**
2015.2

---

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
// Add additional code
...
var assistant = serverWidget.createAssistant({
  title: 'Simple Assistant'
});
assistant.addStep({
  id: 'idname',
  label: 'Sample label'
});
assistant.hideStepNumber = true;
...
// Add additional code
```
Assistant.title

Property Description  The title for the assistant. The title appears at the top of all assistant pages. This value overrides the title specified in `serverWidget.createAssistant(options)`.

Type  string

Supported Script Types  SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (`beforeLoad(scriptContext)`)

Module  N/ui/serverWidget Module

Since  2015.2

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant'
});
var title = assistant.title;
...
//Add additional code
```

serverWidget.AssistantStep

Object Description  Encapsulates a step within a custom NetSuite assistant. Create a step by calling `Assistant.addStep(options)`. For a complete list of this object’s methods and properties, see AssistantStep Object Members.

Supported Script Types  SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (`beforeLoad(scriptContext)`)  

Module  N/ui/serverWidget Module

Since  2015.2
AssistantStep.getFieldIds()

Method Description | Gets the IDs for all the fields in a step.
---|---
Returns | string[]
Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
Governance | None
Module | N/ui/serverWidget Module
Since | 2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
   title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
   id : 'idname',
   label : 'Sample label'
});
...
//Add additional code
var assistant.addField({
   id : 'fieldid',
   type : serverWidget.FieldType.TEXT,
   label : 'Field'
});
```
AssistantStep.getSublistFieldIds(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Gets the IDs for all the sublist fields (line items) in a step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string[]</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.group</td>
<td>string</td>
<td>required</td>
<td>The sublist internal ID.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var sublist = assistant.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Editor'
});
var sublistFieldIds = assistantStep.getSublistFieldIds({
    group : 'sublistid'
});
...
//Add additional code
```
# AssistantStep.getLineCount(options)

## Method Description

Gets the number of lines on a sublist in a step.

**Note:** The first line number on a sublist is 0 (not 1).

## Returns

The count of line items on a sublist as a number

**Note:** if the sublist does not exist, -1 is returned.

## Supported Script Types

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

## Governance

None

## Module

N/ui/serverWidget Module

## Since

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.group</td>
<td>string</td>
<td>required</td>
<td>The sublist internal ID.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant(
    title : 'Simple Assistant'
));
var assistantStep = assistant.addStep(
    id : 'idname',
    label : 'Sample label'
));
var sublist = assistant.addSublist(
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Editor'
));
var numLines = assistantStep.getLineCount(
    group : 'sublistid'
));
...
```
AssistantStep.getSubmittedSublistIds()

Method Description
Gets the IDs for all the sublists submitted in a step.

Returns
string[]

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id : 'idname',
    label : 'Sample label'
});
var sublist = assistant.addSublist({
    id : 'sublistid',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Editor'
});
var submittedSublistId = assistantStep.getSubmittedSublistIds();
...
//Add additional code
```

AssistantStep.getSublistValue(options)

Method Description
Gets the current value of a sublist field (line item) in a step.

Returns
The value of a sublist field as a string

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.group</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the sublist field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for the sublist field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** The first line number on a sublist is 0 (not 1).

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title: 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id: 'idname',
    label: 'Sample label'
});
var sublist = assistant.addSublist({
    id: 'sublistid',
    type: serverWidget.SublistType.INLINEEDITOR,
    label: 'Editor'
});
var sublistfield = sublist.addField({
    id: 'fieldid',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
var sublistvalue = assistantStep.getSublistValue({
    group: 'sublistid',
    id: 'fieldid',
    line: 0
});
...
//Add additional code
```

AssistantStep.getValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gets the current value(s) of a field or multi-select field.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>string[]</th>
</tr>
</thead>
</table>
## Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title: 'Simple Assistant'
});
var assistantStep = assistant.addStep({
    id: 'idname',
    label: 'Sample label'
});
var field = assistant.addField({
    id: 'fieldid',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
var value = assistantStep.getValue({
    id: 'fieldid'
});
...
//Add additional code
```

### AssistantStep.helpText

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The help text for a step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>

**Supported Script Types** SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Module  
N/ui/serverWidget Module

Since  
2015.2
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code...
assistantStep.helpText = 'Help Text Goes Here.;

//Add additional code
```

### AssistantStep.id

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The internal ID of the step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
  id : 'idname',
  label : 'Sample label'
});
var id = assistantStep.id';
...
//Add additional code
```

### AssistantStep.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The label for the step.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>To create a label when the step is first added to the assistant, you can use the Assistant.addStep(options) method.</td>
</tr>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>
Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
---|---
Module | N/ui/serverWidget Module
Since | 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant'
});
var assistantStep = assistant.addStep({
  id : 'idname',
  label : 'Sample label'
});
var label = assistantStep.label;
...
//Add additional code
```

AssistantStep.stepNumber

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates where this step appears sequentially in the assistant.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The index of this step as a number.</td>
</tr>
</tbody>
</table>

ℹ️ **Note:** A sequence of assistant steps starts at 1.

Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
---|---
Module | N/ui/serverWidget Module
Since | 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title : 'Simple Assistant'
});
```
```javascript
var assistantStep = assistant.addStep({
  id : 'idname',
  label : 'Sample label'
});
var stepNum = assistantStep.stepNumber;
...
//Add additional code
```

**serverWidget.Button**

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates button that appears in a UI object.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>To add a button, use <code>Form.addButton(options)</code> or <code>Sublist.addButton(options)</code>. When adding a button to a record or form, consider using a <code>beforeLoad</code> user event script.</td>
</tr>
<tr>
<td></td>
<td>Custom buttons only appear during Edit mode. On records, custom buttons appear to the left of the printer icon.</td>
</tr>
<tr>
<td>Note:</td>
<td>Currently you cannot use SuiteScript to add or remove a custom button to or from the More Actions menu. You can, however, do this using SuiteBuilder point-and-click customization. See the help topic Configuring Buttons and Actions.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var button = form.addButton({
  id : 'buttonid',
  label : 'Test'
});
...
//Add additional code
```

**Button.isDisabled**

| Property Description | Indicates whether a button is grayed-out and disabled. |
The default value is false. If set to true, the button appears grayed-out in the UI and cannot be clicked.

**Note:** This method is not supported for standard NetSuite buttons. This method can be used with custom buttons only.

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var button = form.addButton({
  id : 'buttonid',
  label : 'Test'
});
button.isHidden = true;
...
//Add additional code
```

### Button.isHidden

**Property Description** Indicates whether the button is hidden in the UI.

The default value is false, which means the button is visible.

If set to true, the button is not visible in the UI.

**Note:** This property is supported on custom buttons and on some standard NetSuite buttons. For a list of supported standard buttons, see N/ui/serverWidget Module Script Samples.

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>
### Syntax

#### Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var button = form.addButton({
  id : 'buttonid',
  label : 'Test'
});
button.isHidden = true;
...
//Add additional code
```

---

### Button.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The label for the button.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>You can use this property to rename a button based on context, for example to re-label a button for particular users that are viewing a page.</td>
</tr>
</tbody>
</table>

**Note:** This property is supported on custom buttons and most standard buttons.

- **Type:** string
- **Supported Script Types:** SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
- **Module:** N/ui/serverWidget Module
- **Since:** 2015.2

### Syntax

#### Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var button = form.addButton({
  id : 'buttonid',
  label : 'Test'
});
var label = button.label;
```
serverWidget.Field

**Object Description**
Encapsulates a body or sublist field. Use fields to record or display information specific to your needs.

To add a Field object, use `Assistant.addField(options)`, `Form.addField(options)`, or `Sublist.addField(options)`.

For a complete list of this object's methods and properties, see Field Object Members.

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**
N/ui/serverWidget Module

**Since**
2015.2

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var field = form.addField({
  id : 'custpage_text',
  type : serverWidget.FieldType.TEXT,
  label : 'Text'
});
... //Add additional code
```

Field.addSelectOption(options)

**Method Description**
Adds the select options that appears in the dropdown of a field.

⚠️ **Important:** After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with `Field.addSelectOption(options)`. The select values are determined by the source record or list.

**Returns**
void

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None
## N/ui/serverWidget Module

Since 2015.2

### 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>options.value</td>
<td>string</td>
<td>required</td>
<td>The internal ID of this select option.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.text</td>
<td>string</td>
<td>required</td>
<td>The label for this select option.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.isSelected</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>If set to true, this option is selected by default in the UI. The default value for this parameter is false.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var selectField = form.addField({
    id: 'custpage_selectfield',
    type: serverWidget.FieldType.SELECT,
    label: 'Select'
});

selectField.addSelectOption({
    value: '',
    text: ''
});

selectField.addSelectOption({
    value: 'a',
    text: 'Albert'
});
...
//Add additional code
```

### Field.getSelectOptions(options)

**Method Description**

Obtains a list of available options on a select field.

The internal ID and label of the options for a select field as name/value pairs is returned.
The first 1,000 available options are returned.
If you attempt to get select options on a field that is not a select field, or if you reference a field that does not exist on the form, null is returned.

**Note:** A call to this method may return different results for the same field for different roles.

Returns

Object[]

Supported Script Types

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance

None

Module

N/ui/serverWidget Module

Since

2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.filter</td>
<td>string</td>
<td>optional</td>
<td>A search string to filter the select options that are returned. For example, if there are 50 select options available, and 10 of the options contains 'John', e.g. &quot;John Smith&quot; or &quot;Shauna Johnson&quot;, only those 10 options will be returned. Filter values are case insensitive. The filters 'John' and 'john' will return the same select options.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.#filteroperator</td>
<td>string</td>
<td>optional</td>
<td>Supported operators are contains</td>
<td>is</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var selectField = form.addField({
    id : 'custpage_selectfield',
    type : serverWidget.FieldType.SELECT,
    label : 'Select'
});
selectField.addSelectOption({
    value : 'a',
});
```
Field.setHelpText(options)

**Method Description**
Sets the help text for the field.

When the field label is clicked, a popup displays the help text defined using this method.

**Returns**
The serverWidget.Field object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

### Parameters

- **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.help</td>
<td>string</td>
<td>required</td>
<td>The text in the field help popup.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.showInline</td>
<td>boolean</td>
<td>optional</td>
<td>If set to <code>true</code>, the field help will display inline below the field on the assistant, and in a field help popup. The default value is <code>false</code>, which means the field help appears in a popup when the field label is clicked and does not appear inline.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** The `inline` parameter is available only to `serverWidget.Field` objects that have been added to `serverWidget.createAssistant(options)` objects.

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
```
Field.updateBreakType(options)

Method Description | Updates the break type used to add a break in flow layout for the field.
Returns | serverWidget.Field object
Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
Governance | None
Module | N/ui/serverWidget Module
Since | 2016.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.breakType</td>
<td>serverWidget.FieldBreakType</td>
<td>required</td>
<td>The break type of the field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.setHelpText({
    help : "Help Text Goes Here."
});
...
//Add additional code
```
Field.updateDisplaySize(options)

- **Method Description**: Updates the width and height of the field.
  - Only supported on multi-selects, long text, and fields that get rendered as INPUT (type=text) fields. This function is not supported on list/record fields or rich text fields.
  - To set height and width for rich text fields, use Field.richTextWidth and Field.richTextHeight.

- **Returns**: serverWidget.Field object

- **Supported Script Types**: SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

- **Governance**: None

- **Module**: N/ui/serverWidget Module

- **Since**: 2016.1

**Parameters**

- **Note**: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.height</td>
<td>number</td>
<td>required</td>
<td>The new height of the field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.width</td>
<td>number</td>
<td>required</td>
<td>The new width of the field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

- **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var field = form.addField({
    id: 'custpage_textfield',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
field.updateDisplaySize({
```

---

**SuiteScript 2.0 API Reference**
Field.updateDisplayType(options)

**Method Description**
Updates the display type for the field.

**Returns**
serverWidget.Field object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2016.1

**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>options.displayType</td>
<td>string</td>
<td>required</td>
<td>The new display type of the field. For more information about possible values, see serverWidget.FieldDisplayType.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
   title : 'Simple Form'
});
var field = form.addField({
   id : 'custpage_textfield',
   type : serverWidget.FieldType.TEXT,
   label : 'Text'
});
field.updateDisplayType({
   displayType : serverWidget.FieldDisplayType.HIDDEN
});
...
//Add additional code
```
Field.updateLayoutType(options)

Method Description | Updates the layout type for the field.

Returns | serverWidget.Field object

Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance | None

Module | N/ui/serverWidget Module

Since | 2016.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.layoutType</td>
<td>serverWidget.##FieldLayoutType</td>
<td>required</td>
<td>The new layout type of the field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateLayoutType({
    layoutType : serverWidget.FieldLayoutType.NORMAL
});
...
//Add additional code
```

Field.alias

<table>
<thead>
<tr>
<th>Property Description</th>
<th>An alternate name that you can assign to a serverWidget.Field object. By default, the alias is equal to the field's internal ID. This property is only supported on scripted fields created using the N/ui/serverWidget Module.</th>
</tr>
</thead>
</table>
Field.defaultValue

**Property Description**

The default value for this field.

If you pass an empty string or any value that is not a number, such as `undefined`, the field defaults to a blank field in the UI.

This property is supported only on scripted fields created using the `N/ui/serverWidget Module`.

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/ui/serverWidget Module Script Samples`.

```javascript
//Add additional code
...
var form = serverWidget.createForm(
    title : 'Simple Form'
);
var field = form.addField(
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
);
field.alias = 'fieldId';
...
//Add additional code
```
```javascript
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var field = form.addField({
    id: 'custpage_textfield',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
field.defaultValue = 'Insert Text Here.';
...
//Add additional code
```

**Field.id**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The field internal ID.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var field = form.addField({
    id: 'custpage_textfield',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
var fieldId = field.id;
...
//Add additional code
```

**Field.isMandatory**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether the field is mandatory or optional. If set to true, then the field is defined as mandatory. The default value is false. This property is supported only on scripted fields created using the N/ui/serverWidget Module.</th>
</tr>
</thead>
</table>
### Field.label

| Property Description | The field label.  
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code  
...  
var form = serverWidget.createForm({  
    title : 'Simple Form'  
});  
var field = form.addField({  
    id : 'custpage_textfield',  
    type : serverWidget.FieldType.TEXT,  
    label : 'Text'  
});  
field.isMandatory = true;  
...  
//Add additional code
```
```javascript
var field = form.addField({
  id : 'custpage_textfield',
  type : serverWidget.FieldType.TEXT,
  label : 'Text'
});
var label = field.label;
...
//Add additional code
```

**Field.linkText**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The text displayed for a link in place of the URL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var field = form.addField({
  id : 'custpage_textfield',
  type : serverWidget.FieldType.URL,
  label : 'URL'
});
field.linkText = 'NetSuite';
...
//Add additional code
```

**Field.maxLength**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The maximum length, in characters, of the field (only valid for text, rich text, long text, and textarea fields). This property is supported only on scripted fields created using the N/ui/serverWidget Module.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
</tbody>
</table>
Field.padding

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The number of empty vertical character spaces above the field. This property is supported only on scripted fields created using the N/ui/serverWidget Module.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.maxLength = 64;
...
//Add additional code
```
Field.richTextHeight

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The height of a rich text field, in pixels. The minimum value is 100 pixels and the maximum value is 500 pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```
//Add additional code
...
var form = serverWidget.createForm({
   title: 'Simple Form'
});
var field = form.addField({
   id: 'custpage_textfield',
   type: serverWidget.FieldType.RICHTEXT,
   label: 'Rich Text'
});
field.richTextHeight = 50;
...
//Add additional code
```

Field.richTextWidth

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The width of a rich text field, in pixels. The minimum value is 250 pixels and the maximum value is 800 pixels.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>number</td>
</tr>
</tbody>
</table>
**Supported Script Types**

| Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext)) |

**Module**

| Module | N/ui/serverWidget Module |

**Since**

| Since | 2015.2 |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var field = form.addField(
  id : 'custpage_textfield',
  type : serverWidget.FieldType.RICHTEXT,
  label : 'Rich Text'
);
field.richTextWidth = 100;
...
//Add additional code
```

**Field.type**

| Property Description | The field type. For example, text, date, currency, select, checkbox etc.
The maximum character limit for select field types is 801. |

| Type | string (read-only) |

**Supported Script Types**

| Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext)) |

**Module**

| Module | N/ui/serverWidget Module |

**Since**

| Since | 2015.2 |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
```
var field = form.addField({
  id : 'custpage_textfield',
  type : serverWidget.FieldType.TEXT,
  label : 'Text'
});
var fieldtype = field.type;
...
//Add additional code

serverWidget.FieldGroup

Object Description  Encapsulates a field group on serverWidget.createAssistant(options) objects and on serverWidget.Form objects. A field group is a collection of fields that can be displayed in a one or two column format. Assign a field to a field group in order to label, hide or collapse a group of fields.

For a complete list of this object's properties, see FieldGroup Object Members.

Supported Script Types  SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Module  N/ui/serverWidget Module

Since  2015.2

Syntax

Important:  The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
  id : 'fieldgroupid',
  label : 'Field Group'
});
var field = form.addField({
  id : 'custpage_textfield',
  type : serverWidget.FieldType.TEXT,
  label : 'Text',
  container : 'fieldgroupid'
});
...
//Add additional code

FieldGroup.isBorderHidden

Property Description  Indicates whether the field group can be collapsed.
FieldGroup.isCollapsible

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether the field group can be collapsed.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The default value is <strong>false</strong>, which means the field group displays as a static group that cannot be opened or closed.</td>
</tr>
<tr>
<td></td>
<td>If set to true, the field group can be collapsed.</td>
</tr>
<tr>
<td></td>
<td>Only supported for fields on <code>serverWidget.createAssistant(options)</code> objects</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>boolean <code>true</code></th>
<th><code>false</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td><code>SuiteScript 2.0 Suitelet Script Type</code> and <code>SuiteScript 2.0 User Event Script Type</code> (beforeLoad(scriptContext))</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td><code>N/ui/serverWidget Module</code></td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/ui/serverWidget Module Script Samples`.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
  id : 'fieldgroupid',
  label : 'Field Group'
});
var field = form.addField({
  id : 'custpage_textfield',
  type : serverWidget.FieldType.TEXT,
  label : 'Text',
  container : 'fieldgroupid'
});
fieldgroup.isBorderHidden = true;
...
//Add additional code
```
FieldGroup.isCollapsed

**Property Description**
Indicates whether field group is collapsed or expanded.

The default value is *false*, which means that when the page loads, the field group will not appear collapsed.

If set to true, the field group is collapsed.

Only supported for fields on `serverWidget.createAssistant(options)` objects.

**Type**
boolean `true` | `false`

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**
`N/ui/serverWidget Module`

**Since**
2015.2

### Syntax

![Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/ui/serverWidget Module Script Samples.`](image)

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
fieldgroup.isCollapsible = true;
...
//Add additional code
```
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
var aligned = fieldgroup.isSingleColumn;

FieldGroup.isSingleColumn

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Indicates whether the field group is aligned. The default value is false.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>boolean true</td>
</tr>
</tbody>
</table>

**Supported Script Types**: SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**: N/ui/serverWidget Module

**Since**: 2015.2

**Syntax**

⚠️ **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});
var field = form.addField({
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
});
var aligned = fieldgroup.isSingleColumn;
```
FieldGroup.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The label for the field group.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm(
    title : 'Simple Form'
);  
var fieldgroup = form.addFieldGroup({
    id : 'fieldgroupid',
    label : 'Field Group'
});  
var field = form.addField(
    id : 'custpage_textfield',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    container : 'fieldgroupid'
);  
var label = fieldgroup.label;
...
//Add additional code
```

serverWidget.Form

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates a NetSuite-looking form</th>
</tr>
</thead>
</table>

After you create a `Form` object, you can:

- Add a variety of scriptable elements to the form including fields, links, buttons, tabs, and sublists.

For a complete list of this object's methods and properties, see [Form Object Members](#).
Form.addButton(options)

Method Description
Adds a button to a form.

Returns
serverWidget.Button object

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of the button. If you are adding the button to an existing page, the internal ID must be in lowercase, contain no spaces, and include the prefix custpage. For example, if you add a button that appears as Update Order, the button internal ID should be something similar to custpage_updateorder.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this button.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Form.addCredentialField(options)

### Method Description
Adds a text field that lets you store credentials in NetSuite to be used when invoking services provided by third parties. The GUID generated by this field can be reused multiple times until the script executes again.

For example, when executing credit card transactions, merchants need to store credentials in NetSuite that are used to communicate with Payment Gateway providers.

The credentials added with this method can be used with the N/sftp Module and the N/https Module.

Note the following about this method:

- Credentials associated with this field are stored in encrypted form.
- No piece of SuiteScript holds a credential in clear text mode.
- NetSuite reports or forms will never provide to the end user the clear text form of a credential.
- Any exchange of the clear text version of a credential with a third party must occur over SSL.
- For no reason will NetSuite ever log the clear text value of a credential (for example, errors, debug message, alerts, system notes, and so on).
- Decryption occurs though the scripts listed in the restrictToScriptIds parameter. These scripts can call https.createSecureString(options) to decrypt the GUID and create a SecureString instance.

#### Important
The default maximum length for a credential field is 32 characters. If needed, use the Field.maxLength property to change this value.
**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the credential field. The internal ID must be in lowercase, contain no spaces, and include the prefix custpage if you are adding the field to an existing page.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for the credential field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.restrictDomains</td>
<td>string[]</td>
<td>required</td>
<td>The domains that the credentials can be sent to, such as 'www.mysite.com'. Credentials cannot be sent to a domain that is not specified here. This value can be a domain or a list of domains to which the credentials can be sent.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.restrictIDs</td>
<td>string[]</td>
<td>required</td>
<td>The IDs of the scripts that are allowed to use this credential field. For example, 'customscript_my_script'. Scripts defined here can call https.createSecureString(options) to decrypt the GUID.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.restrictCurrentUser</td>
<td>boolean</td>
<td>optional</td>
<td>Controls whether use of this credential is restricted to the same user that originally entered the credential. By default, the value is false, which means that multiple users can use the credential. For example, multiple clerks at a store making secure calls to a credit processor using a credential that represents the company they work for. If set to true, the credentials apply to a single user.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.container</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of the tab or field group to add the credential field to. By default, the field is added to the main section of the form.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete N/ui/serverWidget Module script example, see N/ui/serverWidget Module Script Samples. For a complete script sample that uses Form.addCredentialField, see N/https Module Script Sample.

```javascript
//Add additional code
...
```

---

**SuiteScript 2.0 API Reference**
```javascript
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var credField = form.addCredentialField(
    {id: 'username',
     label: 'Username',
     restrictToDomains: 'www.mysite.com',
     restrictToScriptIds: 'customscript_my_script',
     restrictToCurrentUser: false,
    });
credField.maxLength = 64;
...
//Add additional code
```

### Form.addField(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a field to a form.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>serverWidget.Field object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the field. The internal ID must be lowercase, contain no spaces, and include the prefix custpage if you are adding the field to an existing page. For example, if you add a field that appears as Purchase Details, the field internal ID should be something similar to custpage_purchasedetails or custpage_purchase_details.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The field type for the field. Use the serverWidget.FieldType enum to define the field type.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Important:** Long text fields created with SuiteScript have a character limit of 100,000. Long text fields created with Suitebuilder have a character limit of 1,000,000.
### Form.addFieldGroup(options)

**Method Description**

Adds a group of fields to a form.

**Returns**

serverWidget.FieldGroup object

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2

### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.container</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of the tab or field group to add the field to. By default, the field is added to the main section of the form.</td>
</tr>
</tbody>
</table>

**Important:** After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with `Field.addSelectOption(options)`. The select values are determined by the source record or list.

**Note:** For radio fields only, the source parameter must contain the internal ID for the field. For more information about working with radio buttons, see the help topic Working with Radio Buttons.

**Syntax**

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var field = form.addField({
  id : 'custpage_abc_text',
  type : serverWidget.FieldType.TEXT,
  label : 'Text'
});
...
//Add additional code
```
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>An internal ID for the field group.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this field group.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.tab</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of the tab to add the field group to. By default, the field group is added to the main section of the form.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var fieldgroup = form.addFieldGroup({
  id : 'fieldgroupid',
  label : 'Field Group'
});
var field = form.addField({
  id : 'custpage_text',
  type : serverWidget.FieldType.TEXT,
  label : 'Text',
  container : 'fieldgroupid'
});
...
//Add additional code
```

**Form.addPageInitMessage(options)**

**Method Description**
Shows a message when users view a record or Suitelet. User event context can be used to control whether the message is shown on records in view, create, or edit mode (not applicable for Suitelets). See the help topic `context.UserEventType`.

**Returns**
Void

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2018.2
Parameters

The options object passed to the Form.addPageInitMessage(options) method takes a single property; either a message.Message object, or the same options object that can be passed to the message.create(options) method. The following tables list the parameters for the previously mentioned object property possibilities.

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.message</td>
<td>message.Message</td>
<td>required</td>
<td>Encapsulates the message to be shown on the form.</td>
<td>2018.2</td>
</tr>
<tr>
<td>options.type</td>
<td>message.Type</td>
<td>required</td>
<td>The message type.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.title</td>
<td>string</td>
<td>optional</td>
<td>The message title. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.message</td>
<td>string</td>
<td>optional</td>
<td>The content of the message. This value defaults to an empty string.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.duration</td>
<td>int</td>
<td>optional</td>
<td>The amount of time, in milliseconds, to show the message. The default is 0, which shows the message until Message.hide() is called. If you specify a duration for message.create() and message.show(), the value from the message.show() method call takes precedence.</td>
<td>2018.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...

// Options object as parameter
form.addPageInitMessage({type: message.Type.INFORMATION, message: 'Hello world!', duration: 5000});

// Message object as parameter
var messageObj = message.create({type: message.Type.INFORMATION, message: 'Hello world!', duration: 5000});
form.addPageInitMessage({message: messageObj});

// Show message when the record is in view mode
function beforeLoad(context) {
    if(context.type === 'view')
        context.form.addPageInitMessage(messageOptions);
```
Form.addPageLink(options)

Method Description | Adds a link to a form.
--- | ---

**Note:** You cannot choose where the page link appears.

Returns | void
--- | ---

Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
--- | ---

Governance | None
--- | ---

Module | N/ui/serverWidget Module
--- | ---

Since | 2015.2
--- | ---

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.title</td>
<td>string</td>
<td>required</td>
<td>The text label for the link.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The type of page link to add. Use the <code>serverWidget/FormPageLinkType</code> enum to set the value.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The URL for the link.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...

var form = serverWidget.createForm({
  title : 'Simple Form'
});
form.addPageLink({
  type : serverWidget.FormPageLinkType.CROSSLINK,
  title : 'NetSuite',
  url : 'http://www.netsuite.com'
});
...
//Add additional code
```
Form.addResetButton(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a reset button to a form. The reset buttons allows a user to clear the entries.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>serverWidget.Button object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.label</td>
<td>string</td>
<td>optional</td>
<td>The label used for this button. If no label is provided, the label defaults to Reset.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
form.addResetButton({
  label : 'Reset Button'
});
...
//Add additional code
```

Form.addSecretKeyField(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a secret key field to the form. This key can be used in crypto modules to perform encryption or hashing.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>serverWidget.Field object</td>
</tr>
</tbody>
</table>

**Important:** The default maximum length for a secret key field is 32 characters. If needed, use the Field.maxLength property to change this value.
**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2016.1

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the secret key field. The internal ID must be in lowercase, contain no spaces, and include the prefix custpage if you are adding the field to an existing page.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.restrictToScriptIds</td>
<td>string or string[]</td>
<td>required</td>
<td>The script ID of the script that is allowed to use this field.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The UI label for the field.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.restrictToCurrentUser</td>
<td>boolean</td>
<td>optional</td>
<td>Controls whether use of this secret key is restricted to the same user that originally entered the key. By default, the value is false, which means that multiple users can use the key.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.container</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of the tab or field group to add the field to. By default, the field is added to the main section of the form.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete N/ui/serverWidget Module script example, see N/ui/serverWidget Module Script Samples. For a complete script example that uses Form.addSecretKeyField(options), see N/crypto Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
   title : 'Simple Form'
});
skField = form.addSecretKeyField({
   id : 'password',
   restrictToScriptIds : 'customscript_my_script',
   restrictToCurrentUser : false,
});
```
## Form.addSublist(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Add a sublist to a form.</th>
</tr>
</thead>
</table>

**Important:** If the row count exceeds 25, sorting is not supported on static sublists created using this method.

**Returns**
A serverWidget.Sublist object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the sublist. The internal ID must be in lowercase, contain no spaces, and include the prefix custpage if you are adding the sublist to an existing page. For example, if you add a sublist that appears as Purchase Details, the sublist internal ID should be something equivalent to custpage_purchasedetails or custpage_purchase_details.</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this sublist.</td>
</tr>
<tr>
<td>options.tab</td>
<td>string</td>
<td>optional</td>
<td>The tab under which to display this sublist. If empty, the sublist is added to the main tab.</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The sublist type. Use the serverWidget.SublistType enum to set the value.</td>
</tr>
</tbody>
</table>

**Since**
2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
```
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var sublist = form.addSublist({
    id: 'sublistid',
    type: serverWidget.SublistType.INLINEEDITOR,
    label: 'Inline Editor Sublist'
});
...
// Add additional code

Form.addSubmitButton(options)

Method Description: Adds a submit button to a form.

Note: If the row count exceeds 25, sorting is not supported on static sublists created using this method.

Returns: serverWidget.Button object

Supported Script Types: SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance: None

Module: N/ui/serverWidget Module

Since: 2016.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.label</td>
<td>string</td>
<td>optional</td>
<td>The label for this button. If no label is provided, the label defaults to “Save”.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
// Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
form.addSubmitButton({
    label: 'Submit Button'
});
...
```
Form.addSubtab(options)

Method Description | Adds a subtab to a form.

**Note:** In order for your subtab to appear on your form, there must be at least one object assigned to the subtab. Otherwise, the subtab will not appear.

**Note:** If you have less than two subtabs on your form, the subtab will not appear. Instead the fields assigned to the tab will appear at the bottom of the form.

Returns | serverWidget.Tab object

Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type

Governance | None

Module | N/ui/serverWidget Module

Since | 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the subtab. The internal ID must be in lowercase, contain no spaces. If you are adding the subtab to an existing page, include the prefix custpage. For example, if you add a subtab that appears as Purchase Details, the subtab internal ID should be something similar to custpage_purchasedetails or custpage_purchase_details.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this subtab.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.tab</td>
<td>string</td>
<td>optional</td>
<td>The tab under which to display this sublist. If empty, the sublist is added to the main tab.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm(
    title : 'Simple Form'
)`
Form.addTab(options)

**Method Description**
Adds a tab to a form.

**Note:** In order for your tab to appear on your form, there must be at least one object assigned to the tab. Otherwise, the tab will not appear.

**Note:** If you have less than two tabs on your form, the tab will not appear. Instead, the fields assigned to the tab will appear at the bottom of the form.

**Returns**
serverWidget.Tab object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the tab. The internal ID must be in lowercase and contain no spaces. If you are adding the tab to an existing page, include the prefix custpage. For example, if you add a subtab that appears as Purchase Details, the subtab internal ID should be something similar to custpage_purchasedetails or custpage_purchase_details.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this tab.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
```
```javascript
... var form = serverWidget.createForm({
    title: 'Simple Form'
});
var tab = form.addTab({
    id: 'tabid',
    label: 'Tab'
});
...
//Add additional code

Form.getButton(options)
```

Form.getButton(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a Button object by internal ID.</td>
<td>serverWidget.Button object</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type</td>
<td>None</td>
<td>N/ui/serverWidget Module</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the button. Internal IDs must be in lowercase and contain no spaces.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
... var form = serverWidget.createForm({
    title: 'Simple Form'
});
var button = form.addButton({
    id: 'buttonid',
    label: 'Test'
});
var button = form.getButton({
```
Form.getField(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a Field object by internal ID.</td>
<td>serverWidget.Field object</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>N/ui/serverWidget Module</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the field. Internal IDs must be in lowercase and contain no spaces.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addField({
    id : 'custpage_text',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
var field = form.getField({
    id : 'textfield'
});
...
//Add additional code
```

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.
**Form.getSublist(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a Sublist object by internal ID.</td>
<td>serverWidget.Sublist object</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
<td>None</td>
<td>N/ui/serverWidget Module</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

*Note:* The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the sublist. Internal IDs must be in lowercase and contain no spaces.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

*Important:* The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
form.addSublist({
    id: 'sublistid',
    type: serverWidget.SublistType.INLINEEDITOR,
    label: 'Inline Editor Sublist'
});
var sublist = form.getSublist({
    id: 'sublistid'
});
...
//Add additional code
```

**Form.getSubtab(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns a subtab by internal ID.</td>
<td>serverWidget.Tab object</td>
</tr>
</tbody>
</table>
Supported Script Types

| Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext)) |

Governance

| Governance | None |

Module

| Module | N/ui/serverWidget Module |

Since

| Since | 2015.2 |

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the subtab. Internal IDs must be in lowercase and contain no spaces.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addSubtab({
    id : 'subtabid',
    label : 'Subtab'
});
var subtab = form.getSubtab({
    id : 'subtabid'
});
...
//Add additional code
```

Form.getTab(options)

| Method Description | Returns a tab object from its internal ID. |

Returns

| Returns | serverWidget.Tab object. |

Supported Script Types

| Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext)) |

Governance

| Governance | None |

Module

| Module | N/ui/serverWidget Module |

Since

| Since | 2016.1 |
## Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the tab to retrieve.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
form.addTab({
    id : 'tabid',
    label : 'Tab'
});
var tab = form.getTab({
    id : 'tabid'
});
...
//Add additional code
```

## Form.getTabs()

**Method Description**

Returns an array that contains all the tabs in a form.

**Returns**

`serverWidget.Tab[]` objects

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm(
```
Form.insertField(options)

**Method Description**
Inserts a field in front of another field.

**Returns**
void

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.field</td>
<td>serverWidget.Field</td>
<td>required</td>
<td>The Field object to insert.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.nextfield</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the field you are inserting a field in front of.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field1 = form.addField({
    id : 'custpage_text',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
```
var field2 = form.addField(
    id: 'custpage_text2',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
));
form.insertField(
    field: field2,
    nextfield: 'textfield1'
));
...

//Add additional code

### Form.insertSublist(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Inserts a sublist in front of another sublist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.sublist</td>
<td>serverWidget.Sublist</td>
<td>required</td>
<td>The Sublist object to insert.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.nextsublist</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the sublist you are inserting a sublist in front of.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).  

```javascript
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var sublist1 = form.addSublist({
    id: 'sublistid1',
    type: serverWidget.SublistType.INLINEEDITOR,
});
```
Form.insertSubtab(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Inserts a subtab in front of another subtab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.subtab</td>
<td>serverWidget.Tab</td>
<td>required</td>
<td>The Subtab object to insert.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.nextsubtab</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the subtab you are inserting a subtab in front of.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
var subtab1 = form.addSubtab({
  id : 'sublistid1',
  type : serverWidget.SublistType.INLINEEDITOR,
  label : 'Inline Editor Sublist'
});
form.insertSublist({
  sublist : subtab1,
  nextsublist : 'sublistid1'
});
...`
Form.insertTab(options)

Method Description
Inserts a tab in front of another tab.

Returns
void

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.tab</td>
<td>serverWidget.Tab</td>
<td>required</td>
<td>The Tab object to insert.</td>
<td>2016.1</td>
</tr>
<tr>
<td>options.nexttab</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the tab you are inserting a tab in front of.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var tab1 = form.addTab({
    id: 'subtabid1',
    label: 'Subtab'
});
var subtab2 = form.addSubtab({
    id: 'subtabid2',
    label: 'Subtab'
});
form.insertSubtab({
    subtab: subtab2,
    nextsubtab: 'subtabid1'
});
...  //Add additional code
```
Form.removeButton(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Removes a button. This method can be used on custom buttons and certain built-in NetSuite buttons. For more information about built-in NetSuite buttons, see the help topic Button IDs.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>[SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))](<a href="https://oracle">https://oracle</a> netsuite documentation) For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the button to remove. See the help topic Button IDs. The internal ID must be in lowercase and contain no spaces.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](https://oracle netsuite documentation).

```javascript
//Add additional code
...
function beforeLoad(context) {
    var yourForm = context.Form;
```
Form.updateDefaultValues(options)

Method Description | Updates the default values of multiple fields on the form.
Returns | void
Supported Script Types | SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
Governance | None
Module | N/ui/serverWidget Module
Since | 2016.1

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>values</td>
<td>object[]</td>
<td>required</td>
<td>An object containing an array of name/value pairs that map field names to field values.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title: 'Simple Form'
});
var field = form.addField({
  id: 'custpage_text',
  type: serverWidget.FieldType.TEXT,
  label: 'Text'
});
form.updateDefaultValues({
  textfield: 'Text Goes Here'
});
...
//Add additional code
```

Form.clientScriptFileId

Property Description | The internal file ID of client script file to be used in this form.
**N/ui/serverWidget Module**

Use this property when attaching an on demand client script to a server-side script.

**Note:** If you deploy a client script to a form using Form.clientScriptFileId or Form.clientScriptModulePath, using the N/log Module adds the logs to the deployment of the parent script. The parent script can be either a beforeLoad user event script or a SuiteScript 2.0 Suitelet Script Type.

<table>
<thead>
<tr>
<th>Type</th>
<th>number</th>
</tr>
</thead>
</table>

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**

N/ui/serverWidget Module

**Since**

2015.2

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY_VALUE_CONFLICT</td>
<td>You attempted to set this value when the Form.clientScriptModulePath property value has already been specified. For more information, see Form.clientScriptModulePath.</td>
</tr>
</tbody>
</table>

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see **N/ui/serverWidget Module Script Samples**.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
form.clientScriptFileId = 32;
...
//Add additional code
```

**Form.clientScriptModulePath**

The relative path to the client script file to be used in this form.

Use this property when attaching an on demand client script to a server-side script.

**Note:** If you deploy a client script to a form using Form.clientScriptFileId or Form.clientScriptModulePath, using the N/log Module adds the logs to the deployment of the parent script. The parent script can be either a beforeLoad user event script or a SuiteScript 2.0 Suitelet Script Type.

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
</table>

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY_VALUE_CONFLICT</td>
<td>You attempted to set this value when the Form.clientScriptFileId property value has already been specified. For more information, see Form.clientScriptFileId.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
objForm.clientScriptModulePath = 'SuiteScripts/formBehavior.js';
...
//Add additional code
```

Form.title

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The title used for the form.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
</tbody>
</table>

Module N/ui/serverWidget Module

Since 2015.2

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var title = form.title;
...
//Add additional code
```
serverWidget.List

Object Description
Encapsulates a list.
For a complete list of this object's methods and properties, see List Object Members.

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Module
N/ui/serverWidget Module

Since
2015.2

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
...
//Add additional code
```

List.addButton(options)

Method Description
Adds a button to a list.

Returns
serverWidget.Button object

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2

Parameters

⚠️ Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the button. The internal ID must be in lowercase, contain no spaces, and include the prefix custpage if you are</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### List.addColumn(options)

**Method Description**

Adds a column to a list.

**Returns**

serverWidget.ListColumn object

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of this column.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
## List.addEditColumn(options)

**Method Description**
Adds a column containing Edit or Edit/View links to a Suitelet or Portlet list. These Edit or Edit/View links appear to the left of a previously existing column.

**Returns**
serverWidget.ListColumn object

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2
Parameters

⚠️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.column</td>
<td>object</td>
<td>required</td>
<td>The Edit/View column is added to the left of the column specified here.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.showHrefCol</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>If set to true, the URL for the link is clickable.</td>
</tr>
<tr>
<td>options.showView</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>If true then an Edit/View column will be added. Otherwise only an Edit column will be added.</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});

var columnId = portlet.addColumn({
    id: 'internalId',
    type: 'text',
    label: 'columnLabel'
});

list.addEditColumn({
    column : columnId,
    showView : true
});
...
//Add additional code
```

**List.addPageLink(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a link to a list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
</tbody>
</table>
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.title</td>
<td>string</td>
<td>required</td>
<td>The text label for the link.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The type of page link to add.</td>
<td>2015.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information about possible values, see serverWidget.FormPageLinkType.</td>
<td></td>
</tr>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The URL for the link.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.addPageLink({
    title : 'NetSuite',
    type : serverWidget.FormPageLinkType.CROSSLINK,
    url : 'http://www.netsuite.com'
});
...
//Add additional code
```

**List.addRow(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a single row to a list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>serverWidget.List</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.row</td>
<td>object</td>
<td>required</td>
<td>A row that consists of either a search.Result, or name/value pairs. Each pair should contain the value for the corresponding Column object in the list.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var list = serverWidget.createList({
  title: 'Simple List'
});
list.addRow({
  row: { columnid1: 'value1', columnid2: 'value2' }
});
...
//Add additional code
```

List.addRows(options)

Method Description
Adds multiple rows to a list.

Returns
serverWidget.ListColumn

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.rows</td>
<td>object</td>
<td>required</td>
<td>An array of rows that consist of either a search.Result array, or an array of name/value pairs.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### List.clientScriptFileId

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>The file cabinet ID of client script file to be used in this list.</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY_VALUE_CONFLICT</td>
<td>You attempted to set this value when the List.clientScriptModulePath property value has already been specified. For more information, see List.clientScriptModulePath.</td>
</tr>
</tbody>
</table>

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
lst.addRows(
  rows : [
    {columnid1 : 'value1', columnid2 : 'value2'},
    {columnid1 : 'value2', columnid2 : 'value3'}
  ]
);
...
//Add additional code
```
var list = serverWidget.createList({
    title : 'Simple List'
});
list.clientScriptFileId = 123;
...
//Add additional code

### List.clientScriptModulePath

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The relative path to the client script file to be used in this list.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.2</td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROPERTY_VALUE_CONFLICT</td>
<td>You attempted to set this value when the List.clientScriptFileId property value has already been specified. For more information, see List.clientScriptFileId.</td>
</tr>
</tbody>
</table>

#### Syntax

```
//Add additional code
...
objList.clientScriptModulePath = 'SuiteScripts/listBehavior.js';
...
//Add additional code
```

### List.style

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Sets the display style for this list. For more information about possible values, see serverWidget.ListStyle.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
</tbody>
</table>
List.title

**Property Description**
Sets the list title.

**Type**
string

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**
N/ui/serverWidget Module

**Since**
2015.2

---

serverWidget.ListColumn

**Object Description**
Encapsulates a list column
For a complete list of this object's methods and properties, see ListColumn Object Members.

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**

N/ui/serverWidget Module

**Since**

2015.2

---

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
var listcolumn = list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.TEXT,
    label : 'Text',
    align : serverWidget.LayoutJustification.RIGHT
});
...
//Add additional code
```

**ListColumn.addParamToURL(options)**

**Method Description**

Adds a URL parameter (optionally defined per row) to the list column's URL.

**Returns**

serverWidget.ListColumn object

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2016.1

---

**Parameters**

⚠️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.param</td>
<td>string</td>
<td>required</td>
<td>The name for the parameter.</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>required</td>
<td>The value for the parameter.</td>
</tr>
</tbody>
</table>
### ListColumn.setURL(options)

**Method Description**
Sets the base URL for the list column.

**Returns**
serverWidget.ListColumn

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2016.1

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.url</td>
<td>string</td>
<td>required</td>
<td>The base URL or a column in the data source that returns the base URL for each row</td>
</tr>
<tr>
<td>options.dynamic</td>
<td>boolean</td>
<td>optional</td>
<td>If true, then the URL is actually an alias that is calculated per row.</td>
</tr>
</tbody>
</table>

---

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var list = serverWidget.createList({
  title : 'Simple List'
});
var listcolumn = list.addColumn({
  id : 'column1',
  type : serverWidget.FieldType.URL,
  label : 'URL',
});
listcolumn.addParamToURL({
  param : 'index',
  value : '3'
});
...
//Add additional code
```
ListColumn.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>This list column label.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
var listcolumn = list.addColumn({
    id : 'column1',
    type : serverWidget.FieldType.URL,
    label : 'URL',
});
listcolumn.setURL({
    url : 'http://www.netsuite.com'
})
...
//Add additional code
var label = listcolumn.label;
```
serverWidget.Sublist

Object Description

Encapsulates a sublist on a serverWidget.Form or an serverWidget.createAssistant(options) object.

To add a sublist, use Assistant.addSublist(options) or Form.addSublist(options).

Note: This object is read-only except for instances created via the serverWidget module using Suitelets or beforeLoad user event scripts.

For a complete list of this object's methods and properties, see Sublist Object Members.

Supported Script Types

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Module

N/ui/serverWidget Module

Since

2015.2

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm(
    title : 'Simple Form'
));
var sublist = form.addSublist(
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
...
//Add additional code
```

Sublist.addButton(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a button to a sublist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>serverWidget.Button</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
</tbody>
</table>
Parameters

- **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the button. The internal ID must be in lowercase and without spaces.</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for the button.</td>
</tr>
<tr>
<td>options.functionName</td>
<td>string</td>
<td>optional</td>
<td>The function name to be triggered on a button click.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.addButton({
    id : 'buttonId',
    label : 'Test'
});
...
//Add additional code
```

**Sublist.addField(options)**

- **Method Description:** Adds a field to a sublist.
- **Returns:** serverWidget.Field object
- **Supported Script Types:** SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
- **Governance:** None
- **Module:** N/ui/serverWidget Module
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID for this field. The internal ID must be in lowercase and without spaces.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.label</td>
<td>string</td>
<td>required</td>
<td>The label for this field.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.type</td>
<td>string</td>
<td>required</td>
<td>The field type. Use the <code>serverWidget.FieldType</code> enum to set this value. The INLINEHTML and value is not supported with this method. The MULTISELECT value is not supported for SuiteScript 2.0 Suitelets.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.source</td>
<td>string</td>
<td>optional</td>
<td>The internalId or scriptId of the source list for this field. Use this parameter if you are adding a select (List/Record) type of field.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Note:** If you have set the `type` parameter to `SELECT`, and you want to add custom options to the select field, you must set `source` to NULL.

**Note:** If you want to add custom options on a select field, you must set the `source` parameter to NULL.

**Important:** After you create a select or multi-select field that is sourced from a record or list, you cannot add additional values with `Field.addSelectOption(options)`. The select values are determined by the source record or list.

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
```
```javascript
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
sublist.addField({
    id : 'fieldId',
    type : serverWidget.FieldType.DATE,
    label : 'Date'
});
...
//Add additional code
```

### Sublist.addMarkAllButtons()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a Mark All and an Unmark All button to a LIST type of sublist.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>A serverWidget.Button[] object</th>
</tr>
</thead>
</table>

**Supported Script Types**

- SuiteScript 2.0 Suitelet Script Type
- SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

N/ui/serverWidget Module

**Since**

2015.2

### Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.LIST,
    label : 'List Sublist'
});
sublist.addMarkAllButtons();
...
//Add additional code
```

### Sublist.addRefreshButton()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Adds a Refresh button to a LIST type of sublist.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
<th>serverWidget.Button object</th>
</tr>
</thead>
</table>
Sublist.getField(options)

**Method Description**
Returns a Field object on a sublist.

**Returns**
serverWidget.Field

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2016.2

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The field internal ID (for example, use item as the ID for the item field). For more information about supported sublists, internal IDs, and field IDs, see the SuiteScript Records Browser.</td>
</tr>
</tbody>
</table>
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var itemField = form.getSublist({id: 'item'}).getField({id: 'item'});
...
//Add additional code
```

### Sublist.getSublistValue(options)

**Method Description**
Gets a field value on a sublist.

**Returns**
string

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**
None

**Module**
N/ui/serverWidget Module

**Since**
2015.2

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID of a field.</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for this field.</td>
</tr>
</tbody>
</table>

**Note:** The first line number on a sublist is 0 (not 1).

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var sublistvalue = sublist.getSublistValue({
    id : 'quantity',
    line: 1
});
```
Sublist.setSublistValue(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Sets the value of a sublist field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>void</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the line item field being set.</td>
</tr>
<tr>
<td>options.line</td>
<td>number</td>
<td>required</td>
<td>The line number for this field.</td>
</tr>
</tbody>
</table>

**Note:** The first line number on a sublist is 0 (not 1).

| options.value | string | required | The value for the field being set. |

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
//Add additional code
var form = serverWidget.createForm({
   title : 'Simple Form'
});
var sublist = form.addSublist({
   id : 'sublist',
   type : serverWidget.SublistType.INLINEEDITOR,
   label : 'Inline Editor Sublist'
});

sublist.addField({
   id : 'sublist',
   type: ui.FieldType.TEXT,
```
Sublist.updateTotallingFieldId(options)

Method Description
Updates the ID of a field designated as a totalling column, which is used to calculate and display a running total for the sublist.

Returns
serverWidget.Sublist object

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.id  | string  | required            | The internal ID name of the field to use as a total field.

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
sublist.updateTotallingFieldId({
  id : 'fieldId'
});
...
//Add additional code
```

Sublist.updateUniqueFieldId(options)

Method Description
Updates a field ID that is to have unique values across the rows in the sublist.
Note: This method is available on inlineeditor and editor sublists only.

Returns
serverWidget.Sublist object

Supported Script Types
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

Governance
None

Module
N/ui/serverWidget Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>The internal ID name of the field to use as a unique field.</td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm(
    title : 'Simple Form'
);
var sublist = form.addSublist(
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
);
sublist.addField(
    id : 'fieldId',
    type : serverWidget.FieldType.DATE,
    label : 'Date'
);
sublist.updateUniqueFieldId(
    id : 'fieldId'
)
...
//Add additional code
```

Sublist.displayType

Property Description
The display style for a sublist.
Use the `serverWidget.SublistDisplayType` enum to set this value.

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td><code>SuiteScript 2.0 Suitelet Script Type</code> and <code>SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</code></td>
</tr>
<tr>
<td>Module</td>
<td><code>N/ui/serverWidget Module</code></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/ui/serverWidget Module Script Samples`.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var sublist = form.addSublist({
    id: 'sublist',
    type: serverWidget.SublistType.INLINEEDITOR,
    label: 'Inline Editor Sublist'
});
sublist.displayType = serverWidget.SublistDisplayType.HIDDEN;
...  
//Add additional code
```

---

### Sublist.helpText

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The inline help text for a sublist.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td><code>SuiteScript 2.0 Suitelet Script Type</code> and <code>SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</code></td>
</tr>
<tr>
<td>Module</td>
<td><code>N/ui/serverWidget Module</code></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see `N/ui/serverWidget Module Script Samples`.

```javascript
//Add additional code
...
var form = serverWidget.createForm(
```
```javascript
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var sublist = form.addSublist({
    id: 'sublist',
    type: serverWidget.SublistType.INLINEEDITOR,
    label: 'Inline Editor Sublist'
});
sublist.helpText = "Help Text Goes Here."
...
//Add additional code
```

### Sublist.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The label for this sublist.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td></td>
</tr>
</tbody>
</table>

**Supported Script Types**

- SuiteScript 2.0 Suitelet Script Type
- SuiteScript 2.0 User Event Script Type (`beforeLoad(scriptContext)`)

**Module**

N/ui/serverWidget Module

**Since**

2015.2

---

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var sublist = form.addSublist({
    id: 'sublist',
    type: serverWidget.SublistType.INLINEEDITOR,
    label: 'Inline Editor Sublist'
});
var label = sublist.label;
...
//Add additional code
```

### Sublist.lineCount

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The number of line items on a sublist.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** The first line number on a sublist is 0 (not 1).

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number (read-only)</td>
<td></td>
</tr>
</tbody>
</table>
### serverWidget.Tab

**Object Description**
Encapsulates a tab or subtab on a `serverWidget.Form` object.

You can add a new tab or subtab to a form using one of the following methods:

- `Form.addSubtab(options)`
- `Form.addTab(options)`
- `Form.insertSubtab(options)`
- `Form.insertTab(options)`

The internal ID must be in lowercase, contain no spaces, and include the prefix `custpage` if you are adding the field to an existing page.

**Note:** In order for your tab to appear on your form, there must be at least one object assigned to the tab. Otherwise, the tab will not appear.

**Note:** If you have less than two tabs on your form, the tab will not appear. Instead the fields assigned to the tab will appear at the bottom of the form.

For a complete list of this object's properties, see Tab Object Members.

**Supported Script Types**
SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**
`N/ui/serverWidget Module`

**Since**
2015.2
Tab.helpText

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The inline help text for a tab or subtab.</td>
</tr>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```
//Add additional code
...
var form = serverWidget.createForm(
    title: 'Simple Form'
);
var tab = form.addTab(
    id: 'tabid1',
    label: 'Tab 1'
);
var tab = form.addTab(
    id: 'tabid2',
    label: 'Tab 2'
);
form.addField(
    id: 'custpage_tabid1',
    type: ui.FieldType.TEXT,
    label: 'Tab 1 Field'
);
form.addField(
    id: 'custpage_tabid2',
    type: ui.FieldType.TEXT,
    label: 'Tab 2 Field'
)}...
//Add additional code
```
... var form = serverWidget.createForm({
  title: 'Simple Form'
});
var tab = form.addTab({
  id: 'tabid',
  label: 'Tab'
});
tab.helpText = 'Help Text Goes Here';
...
//Add additional code

Tab.label

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The label for a tab or subtab.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title: 'Simple Form'
});
var tab = form.addTab({
  id: 'tabid',
  label: 'Tab'
});
var label = tab.label;
...
//Add additional code
```

serverWidget.createAssistant(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates an assistant object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>serverWidget.Assistant object</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</td>
</tr>
</tbody>
</table>
**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>options.title</td>
<td>string</td>
<td>required</td>
<td>The title of the assistant. This title appears at the top of all assistant pages.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.hideNavBar</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>Indicates whether to hide the navigation bar menu.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>By default, set to false. The header appears in the top-right corner on the assistant.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If set to true, the header on the assistant is hidden from view.</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
  title: 'Simple Assistant'
});
...
//Add additional code
```

serverWidget.createForm(options)

**Method Description**

Creates a form object.

**Returns**

`serverWidget.Form` object

**Supported Script Types**

*SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Governance**

None

**Module**

`N/ui/serverWidget Module`

**Since**

2015.2
**Parameters**

> **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.title</td>
<td>string</td>
<td>required</td>
<td>The title of the form.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.hideNavBar</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>Indicates whether to hide the navigation bar menu. By default, set to false. The header appears in the top-right corner on the form. If set to true, the header on the assistant is hidden from view.</td>
</tr>
</tbody>
</table>

**Syntax**

> **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code ...
var form = serverWidget.createForm({
  title : 'Simple Form'
});
...
//Add additional code
```

serverWidget.createList(options)

- **Method Description**: Instantiates a standalone list.
- **Returns**: serverWidget.List object
- **Supported Script Types**: SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
- **Governance**: None
- **Module**: N/ui/serverWidget Module
- **Since**: 2015.2

**Parameters**

> **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.title</td>
<td>string</td>
<td>required</td>
<td>The title of the list.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>
### 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>options.hideNavBar</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>Indicates whether to hide the navigation bar menu. By default, set to false. The header appears in the top-right corner on the form. If set to true, the header on the assistant is hidden from view.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var list = serverWidget.createList({
  title : 'Simple List'
});
...
//Add additional code
```

### serverWidget.AssistantSubmitAction

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
</table>
|                 | Holds the string values for submit actions performed by the user. This enum is used to set the value of the `Assistant.getLastAction()`.
|                 | **After a finish action is submitted, by default, the text "Congratulations! You have completed the <assistant title>" appears on the finish page.**
|                 | In a non-sequential process (steps are unordered), `jump` is used to move to the user's last action. |

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

### Supported Script Types

- SuiteScript 2.0 Suitelet Script Type
- SuiteScript 2.0 User Event Script Type
- `beforeLoad(scriptContext)`

### Module

- N/ui/serverWidget Module

### Since

- 2015.2

### Values

- BACK
Syntax

The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var assistant = serverWidget.createAssistant({
    title : 'Simple Assistant'
});
...
if (assistant.getLastAction() == serverWidget.AssistantSubmitAction.CANCEL) {
    ...
}
...
//Add additional code
```

serverWidget.FieldBreakType

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NONE</td>
<td>This is the default value for field break type.</td>
</tr>
<tr>
<td>STARTCOL</td>
<td>This value moves the field into a new column. Additionally, it disables automatic field balancing if set on any field.</td>
</tr>
<tr>
<td>STARTROW</td>
<td>This value places a field located outside of a field group on a new row. This value only works on fields with a Field Layout Type set to OUTSIDE, OUTSIDEABOVE or OUTSIDEbelow. For more information, see serverWidget.FieldLayoutType and Field.updateLayoutType(options).</td>
</tr>
</tbody>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'custpage_text',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
field.updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.OUTSIDE
});
field.updateBreakType({
    breakType : serverWidget.FieldBreakType.STARTROW
});
...
//Add additional code
```

serverWidget.FieldDisplayType

**Enum Description**

Enumeration that holds the string values for supported field display types. This enum is used to set the value of the `displayType` parameter when `Field.updateDisplayType(options)` is called.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**

N/ui/serverWidget Module

**Since**

2015.2

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description of Field Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISABLED</td>
<td>Prevents a user from changing the field</td>
</tr>
<tr>
<td>ENTRY</td>
<td>The sublist field appears as a data entry input field (for a select field without a checkbox)</td>
</tr>
<tr>
<td>HIDDEN</td>
<td>The field on the form is hidden.</td>
</tr>
<tr>
<td>Value</td>
<td>Description of Field Type</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>INLINE</td>
<td>The field appears as inline text</td>
</tr>
<tr>
<td>NORMAL</td>
<td>The field appears as a normal input field (for non-sublist fields)</td>
</tr>
<tr>
<td>READONLY</td>
<td>The field is disabled but it is still selectable and scrollable (for textarea fields)</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippets show the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var field = form.addField({
    id: 'custpage_text',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
field.updateDisplayType({
    displayType: serverWidget.FieldDisplayType.HIDDEN
});
...
//Add additional code
```

### serverWidget.FieldLayoutType

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration that holds the string values for the supported types of field layouts. This enum is used to set the value of the <code>layoutType</code> parameter when <code>Field.updateLayoutType(options)</code> is called.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STARTROW</td>
<td>This value makes the field appear first in a horizontally aligned field group in the normal field layout.</td>
</tr>
<tr>
<td>MIDROW</td>
<td>This value makes the field appear in the middle of a horizontally aligned field group in the normal field layout.</td>
</tr>
<tr>
<td>Enum</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ENDROW</td>
<td>This value makes the field appear last in a horizontally aligned field group in the normal field layout.</td>
</tr>
<tr>
<td>OUTSIDE</td>
<td>This value makes the field appear outside (above or below based on form default) the normal field layout area.</td>
</tr>
<tr>
<td>OUTSIDEBELOW</td>
<td>This value makes the field appear below the normal field layout area. Using this allows you to position a field below a field group.</td>
</tr>
<tr>
<td>OUTSIDEABOVE</td>
<td>This value makes the field appear above the normal field layout area. Using this allows you to position a field above a field group.</td>
</tr>
<tr>
<td>NORMAL</td>
<td>This value makes the fields appear in its default position.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important**: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title: 'Simple Form'
});
var field = form.addField({
    id: 'custpage_text',
    type: serverWidget.FieldType.TEXT,
    label: 'Text'
});
field.updateLayoutType({
    layoutType: serverWidget.FieldLayoutType.OUTSIDEBELOW
});
...
//Add additional code
```

serverWidget.FieldType

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the values for supported field types. This enum is used to set the value of the <code>type</code> parameter when <code>Form.addField(options)</code> is called.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note</strong>:</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
<tr>
<td><strong>Important</strong>:</td>
<td>Long text fields created with SuiteScript have a character limit of 100,000. Long text fields created with Suitebuilder have a character limit of 1,000,000.</td>
</tr>
</tbody>
</table>

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))
Module N/ui/serverWidget Module
Since 2015.2

Values

- CHECKBOX
- CURRENCY
- DATE
- DATETIME
- DATEDT IMETZ
- EMAIL
- FILE
- FLOAT
- HELP
- INLINEHTML
- INTEGER
- IMAGE
- LABEL
- LONGTEXT
- MULTISELECT
- PASSWORD
- PERCENT
- PHONE
- SELECT
- RADIO
- RICHTEXT
- TEXT
- TEXTAREA
- TIMEOFDAY
- URL

Consider the following as you work with these field types:

- The FILE field type is available only for Suitelets and will appear on the main tab of the Suitelet page. FILE fields cannot be added to tabs, subtabs, sublists, or field groups and are not allowed on existing pages.
- The INLINEHTML field type is not supported with Sublist.addField(options).
- The IMAGE field type is available only for fields that appear on list/staticlist sublists. You cannot specify an IMAGE field on a form.
- The MULTISELECT field type is not supported by SuiteScript 2.0 Suitelets.

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form'
});
var field = form.addField({
    id : 'custpage_text',
    type : serverWidget.FieldType.TEXT,
    label : 'Text'
});
...
//Add additional code
```
serverWidget.FormPageLinkType

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration that holds the string values for supported page link types on a form. This enum is used to set the value of the type parameter when Form.addPageLink(options) is called.</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types:** SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module:** N/ui/serverWidget Module

**Since:** 2015.2

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>BREADCRUMB</td>
<td>Link appears on the top-left corner after the system bread crumbs</td>
</tr>
<tr>
<td>CROSSLINK</td>
<td>Link appears on the top-right corner</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title: 'Simple Form'
});
form.addPageLink({
  type: serverWidget.FormPageLinkType.CROSSLINK,
  title: 'NetSuite',
  url: 'http://www.netsuite.com'
});
...
//Add additional code
```

serverWidget.LayoutJustification

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration that holds the string values for supported justification layouts. This enum is used to set the value of the align parameter when List.addColumn(options) is called.</td>
<td></td>
</tr>
</tbody>
</table>

**Supported Script Types:** SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module:** N/ui/serverWidget Module

**Since:** 2015.2
**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

---

**Supported Script Types**

SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))

**Module**

N/ui/serverWidget Module

**Since**

2015.2

---

**Values**

- CENTER
- LEFT
- RIGHT

---

**Syntax**

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var list = serverWidget.createList({
    title: 'Simple List'
});
list.addColumn({
    id: 'column1',
    type: serverWidget.FieldType.TEXT,
    label: 'Text',
    align: serverWidget.LayoutJustification.RIGHT
});
...
//Add additional code
```

---

**serverWidget.ListStyle**

**Enum Description**

Enumeration that holds the string values for supported list styles. This enum is used to set the value of the **List.style** property.

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
Module | N/ui/serverWidget Module
---|---
Since | 2015.2

Values
- GRID
- REPORT
- PLAIN
- NORMAL

Syntax

```javascript
//Add additional code
...
var list = serverWidget.createList({
    title : 'Simple List'
});
list.style = serverWidget.ListStyle.REPORT;
...
//Add additional code
```

serverWidget.SublistDisplayType

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for supported sublist display types. This enum is used to set the value of the Sublist.displayType property.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note</td>
<td>JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
<th>SuiteScript 2.0 Suitelet Script Type and SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>N/ui/serverWidget Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Values
- HIDDEN
- NORMAL
Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/ui/serverWidget Module Script Samples](#).

```javascript
//Add additional code
...
var form = serverWidget.createForm({
  title: 'Simple Form'
});
var sublist = form.addSublist({
  id: 'sublist',
  type: serverWidget.SublistType.INLINEEDITOR,
  label: 'Inline Editor Sublist'
});
sublist.displayType = serverWidget.SublistDisplayType.HIDDEN;
...
//Add additional code
```

serverWidget.SublistType

<table>
<thead>
<tr>
<th>Enum Description</th>
<th>Enumeration that holds the string values for valid sublist types. This enum is used to define the <code>type</code> parameter when <code>Form.addSublist(options)</code> is called</th>
</tr>
</thead>
</table>

**Note:** JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.

**Supported Script Types**

- [SuiteScript 2.0 Suitelet Script Type](#) and [SuiteScript 2.0 User Event Script Type (beforeLoad(scriptContext))](

**Module**

- [N/ui/serverWidget Module](#)

**Since**

- 2015.2

**Values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INLINEDITOR</td>
<td>These types of sublists are both fully editable. The only difference between these types is their appearance in the UI:</td>
</tr>
<tr>
<td></td>
<td>- With an inline editor sublist, a new line is displayed at the bottom of the list after existing lines. To add a line, a user working in the UI clicks inside the new line and adds a value to each column as appropriate. Examples of this style include the Item sublist on the sales order record and the Expense sublist on the expense report record.</td>
</tr>
<tr>
<td></td>
<td>- With an editor sublist, a user in the UI adds a new line by working with fields that are displayed above the existing sublist lines. This style is not common on standard NetSuite record types.</td>
</tr>
<tr>
<td>LIST</td>
<td>This type of sublist has a fixed number of lines. You can update an existing line, but you cannot add lines to it.</td>
</tr>
</tbody>
</table>
### Value Description

**Note:** To make a field within a LIST type sublist editable, use `Field.updateDisplayType(options)` and the enum `serverWidget.FieldDisplayType` to update the field display type to ENTRY.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STATICLIST</td>
<td>This type of sublist is read-only. It cannot be edited in the UI, and it is not available for scripting.</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/ui/serverWidget Module Script Samples.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form with Inline Editor type Sublist'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.INLINEEDITOR,
    label : 'Inline Editor Sublist'
});
...
//Add additional code
```

The following code snippet shows how to make a field within a LIST type sublist editable by updating the `fieldDisplayType` to ENTRY.

```javascript
//Add additional code
...
var form = serverWidget.createForm({
    title : 'Simple Form with List type Sublist'
});
var sublist = form.addSublist({
    id : 'sublist',
    type : serverWidget.SublistType.LIST,
    label : 'List Type Sublist'
});
var internalId = sublist.addField({
    id : 'id',
    label : 'Internal ID',
    type : serverWidget.FieldType.TEXT
});
internalId.updateDisplayType({displayType: serverWidget.FieldDisplayType.ENTRY});
...
//Add additional code
```

### N/url Module

Use the url module to determine URL navigation paths within NetSuite and format URL strings.
N/url Module Members

N/url Module Script Samples

N/url Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>url.format(options)</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Converts (serializes) URL query parameters into a string.</td>
</tr>
<tr>
<td></td>
<td>url.#resolveDomain#</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Returns a domain name for a NetSuite account.</td>
</tr>
<tr>
<td></td>
<td>url.#resolveRecord#</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns an internal URL string to a NetSuite record.</td>
</tr>
<tr>
<td></td>
<td>url.#resolveScript#</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns an external or internal URL string to a script.</td>
</tr>
<tr>
<td></td>
<td>url.#resolveTaskLink#</td>
<td>string</td>
<td>Server-side scripts</td>
<td>Returns an internal URL for a tasklink.</td>
</tr>
</tbody>
</table>

| Enum        | url.HostType              | enum                     | Server-side scripts    | An enum used to populate the hostType parameter of the url.#resolveDomain# method. |

N/url Module Script Samples

The following script samples show how to use the url module.

These samples use the require function, so that you can copy each script into the debugger and test it, after making any necessary edits. Remember that you must use the define function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

**Important:** Some values in these samples are placeholders. Before using these samples, replace this values with valid ones from your NetSuite account. If you run a script with an invalid value, the system may throw an error.

The following example retrieves the relative URL of a record. With the internal ID value used in this example, the returned output would be `/app/accounting/transactions/salesord.nl?id=6&e=T&compid=', followed by the NetSuite account ID.

```javascript
/** *
 * @NApiVersion 2.x
 */
require(['N/url'],
    function(url) {
        var output = url.resolveRecord({
            recordType: 'salesorder',
            recordId: 6,
            isEditMode: true
        });
    });
```
The following example shows how to generate an absolute URL to a specific resource.

```javascript
/**
 * @NApiVersion 2.x
 */
// This example shows how to get the absolute url of a record.
// Company context is required to run this client’s script.

require(['N/url', 'N/record'],
function(url, record) {
    function resolveRecordUrl() {
        var scheme = 'https://';
        var host = url.resolveDomain({
            hostType: url.HostType.APPLICATION
        });
        var relativePath = url.resolveRecord({
            recordType: record.Type.SALES_ORDER,
            recordId: 6,
            isEditMode: true
        });
        var output = scheme + host + relativePath;
    }
    resolveRecordUrl();
});
```

The following example shows how to get the domain for calling a RESTlet.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/url'],
function(url) {
    function resolveDomainUrl() {
        var sCompId = 'MSTRWLF';
        var output = url.resolveDomain({
            hostType: url.HostType.RESTLET,
            accountId: sCompId
        });
    }
    resolveDomainUrl();
});
```

The following example creates a URL and then does a secure HTTPS POST request to that URL with an empty body. The server’s response is logged.

```javascript
/**
 * @NApiVersion 2.x
 */
require(['N/url', 'N/https'], function(url, https) {
    var script = 'customscript1';
    var deployment = 'customdeploy1';
```
## url.format(options)

**Method Description**
Creates a serialized representation of an object containing query parameters.
Use the returned value to build a URL query string.

**Returns**
URL as a string

**Supported Script Types**
All server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/url Module

**Since**
2015.1

### 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>options.domain</td>
<td>string</td>
<td>required</td>
<td>The domain name.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>required</td>
<td>Additional URL parameters as name/value pairs.</td>
<td>2015.1</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see [N/url Module Script Samples](#).

For a script that uses the following code snippet, the returned output is `http://fruitland.com?fruit=grape&seedless=true&variety=Concord+Giant&PLU=4272`, expressed as a string.

```javascript
// Add additional code
...
var output = url.format({
    domain: 'http://fruitland.com',
    params: {
        fruit: 'grape',
        seedless: true,
        variety: 'Concord Giant',
    }
});
```
url.resolveDomain(options)

**Method Description**
Returns a domain name for a NetSuite account.

**Returns**
string

**Supported Script Types**
Client and server-side scripts
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**
None

**Module**
N/url Module

**Since**
2017.1

**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>options.hostType</td>
<td>string</td>
<td>required</td>
<td>The type of domain name you want to retrieve. Set this value using the url.HostType enum.</td>
<td>2017.1</td>
</tr>
<tr>
<td>options.accountId</td>
<td>string</td>
<td>optional</td>
<td>The NetSuite account ID for which you want to retrieve data. If no account is specified, the system returns data on the account that is running the script.</td>
<td>2017.1</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
var output = url.resolveDomain({
  hostType: url.HostType.APPLICATION,
  accountId: '012345'
});
... // Add additional code
```

url.resolveRecord(options)

**Method Description**
Returns the URL string to a NetSuite record.
Returns
URL to a NetSuite record as a string

Supported Script Types
All server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/url Module

Since
2015.1

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.recordType</td>
<td>string</td>
<td>required</td>
<td>The type of record. For example, 'transaction'.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.recordId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the target record instance.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.isEditMode</td>
<td>boolean</td>
<td>required</td>
<td>If set to true, returns a URL for the record in Edit mode.</td>
<td>2015.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If set to false, returns a URL for the record in View mode.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>The default value is View.</td>
<td></td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>Object used to add parameters for a custom URL. For example, a query to a</td>
<td>2015.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>database or to a search engine.</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/url Module Script Samples.

```javascript
//Add additional code
...
var output = url.resolveRecord({
    recordType: 'salesorder',
    recordId: 6,
    isEditMode: true
});
...
//Add additional code

url.resolveScript(options)
```

Method Description
Returns an external or internal URL string to a script.
Returns | The URL as a string
---|---
Supported Script Types | All server-side scripts
Governance | None
Module | N/url Module
Since | 2015.1

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.scriptId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the script. The ID must identify a RESTlet or a Suitelet.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.deploymentId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the deployment script</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.params</td>
<td>Object</td>
<td>optional</td>
<td>The object containing name/value pairs to describe the query.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.returnExternalUrl</td>
<td>boolean</td>
<td>true</td>
<td>optional</td>
<td>Indicates whether to return the external URL. By default, the internal URL is returned.</td>
</tr>
</tbody>
</table>

Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/url Module Script Samples.

```javascript
//Add additional code
...
var output = url.resolveScript({
    scriptId: 'custom_script',
    deploymentId: 'custom_script_deployment',
    returnExternalUrl: true
});
...
//Add additional code
```

`url.resolveTaskLink(options)`

**Method Description** | Returns the internal URL to a NetSuite tasklink.
---|---
**Returns** | The URL as a string
**Supported Script Types** | All server-side scripts
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/url Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.id</td>
<td>string</td>
<td>required</td>
<td>Internal ID for the tasklink.</td>
<td>2015.1</td>
</tr>
<tr>
<td>options.params</td>
<td>Map</td>
<td>optional</td>
<td>The Map object containing name/value pairs to describe the query.</td>
<td>2015.1</td>
</tr>
</tbody>
</table>

Note: Each page in NetSuite has a unique Tasklink Id associated with it for a specific record type. You can determine the Tasklink for a page within NetSuite by viewing the HTML page source. Search for a string similar to the following, where LIST_SCRIPT refers to the TASKLINK: onclick="nlPopupHelp('LIST_#SCRIPT','help').

Syntax

Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/url Module Script Samples.

```javascript
//Add additional code
...
u = url.resolveTaskLink('SRCH_JOB', p);
...
//Add additional code
```

url.HostType

<table>
<thead>
<tr>
<th>Enum</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enumeration whose string values each describe a category of domain name. This enum is used to set the value of the hostType parameter of the url.resolveDomain(options) method.</td>
<td></td>
</tr>
</tbody>
</table>

Note: JavaScript does not include an enumeration type. The SuiteScript 2.0 documentation utilizes the term enumeration (or enum) to describe the following: a plain JavaScript object with a flat, map-like structure. Within this object, each key points to a read-only string value.
N/url Module

For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Module</th>
<th>N/url Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2017.1</td>
</tr>
</tbody>
</table>

Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
<th>Sample Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION</td>
<td>The domain for UI access.</td>
<td>system.na2.netsuite.com</td>
</tr>
<tr>
<td>FORM</td>
<td>The domain for forms hosted online, usually in Suitelets.</td>
<td>forms.na2.netsuite.com</td>
</tr>
<tr>
<td>RESTLET</td>
<td>The domain for calling a RESTlet from an external source.</td>
<td>rest.na2.netsuite.com</td>
</tr>
<tr>
<td>SUITETALK</td>
<td>The domain for SuiteTalk (web services) requests.</td>
<td>webservices.na2.netsuite.com</td>
</tr>
</tbody>
</table>

⚠️ Warning: The results returned, as shown in the sample results column, may change without notice. Because these values can change, your scripts must dynamically discover domain names. For more details, see the help topic Understanding Multiple Data Centers.

Syntax

⚠️ Important: The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/url Module.

```javascript
//Add additional code
...
var output = url.resolveDomain({
  hostType: url.HostType.APPLICATION,
  accountId: '012345'
});
...
//Add additional code
```

N/util Module

This module exposes the util Object and its members, made up primarily of methods that verify type on objects and primitives in a SuiteScript 2.0 script.

Each type verification method (for example, util.isArray(obj)) returns a boolean value, based on evaluation of the obj parameter.

If you need to identify a type specific to SuiteScript 2.0, use the toString() global method.

ℹ️ Note: The util Object can be accessed globally or by loading this module. Load the N/util module when you want to manually access the util module members, such as for testing purposes. For more information about global objects, see SuiteScript 2.0 Global Objects.

- N/util Module Members
- N/util Module Script Sample
<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>util.isArray(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a JavaScript and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.isBoolean(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a Boolean and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.isDate(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a JavaScript Date object and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.each(iterable, callback)</code></td>
<td>Object or Array</td>
<td>Client and server-side scripts</td>
<td>Iterates over each member in an Object or Array.</td>
</tr>
<tr>
<td></td>
<td><code>util.extend(receiver, contributor)</code></td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Copies the properties in a source object to a destination object and returns the destination object.</td>
</tr>
<tr>
<td></td>
<td><code>util.isFunction(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a JavaScript Function object and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.isNumber(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a JavaScript Number object or a value that evaluates to a Number object, and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.isObject(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a strictly a JavaScript Object, and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.isRegExp(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a JavaScript RegExp object or a value that evaluates to a RegExp object, and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.isString(obj)</code></td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the obj parameter is a JavaScript String object or a value that evaluates to a String object, and false otherwise.</td>
</tr>
<tr>
<td></td>
<td><code>util.nanoTime()</code></td>
<td>number</td>
<td>Server-side scripts</td>
<td>Returns the amount of time elapsed from an arbitrary fixed point, in nanoseconds.</td>
</tr>
<tr>
<td></td>
<td><code>util.each(iterable, callback)</code></td>
<td>Object or Array</td>
<td>Client and server-side scripts</td>
<td>Iterates over each member in an Object or Array.</td>
</tr>
<tr>
<td></td>
<td><code>util.extend(receiver, contributor)</code></td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Copies the properties in a source object to a destination object.</td>
</tr>
</tbody>
</table>

### N/util Module Script Sample

For help with writing scripts in SuiteScript 2.0, see the help topics [SuiteScript 2.0 Hello World](#) and [SuiteScript 2.0 Entry Point Script Creation and Deployment](#).

```javascript
require(['N/record'], function(record){
  // Your code here
});
```
// Create a sales order
var rec = record.create({
    type: 'salesorder',
    isDynamic: true
});
rec.setValue(
    fieldId: 'entity',
    value: 107
});

// Set up an object containing an item's internal id and the corresponding quantity
var itemList = {
    39: 5,
    38: 1
}

// Iterate through the object and set the key-value pairs on the record
util.each(itemList, function(quantity, itemId) {
    rec.selectNewLine('item');
    rec.setCurrentSublistValue('item', 'item', itemId);
    rec.setCurrentSublistValue('item', 'quantity', quantity);
    rec.commitLine('item');
});

// log.debug(rec) //Shows the JSON representation of the current values in a record object
var id = rec.save();

---

**Warning:** This script sample includes hard-coded values for the purpose of illustration. To run this sample in the SuiteScript debugger, you must replace these hard-coded values with values from records in your account. For information about debugging, see the help topic Using the SuiteScript Debugger.

---

**util.isArray(obj)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns true if the <code>obj</code> parameter is a JavaScript Array object and false otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>boolean true</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None |
| Module             | N/util Module |
| Global object      | util Object |
| Since              | 2016.1 |

**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>obj</td>
<td>Object</td>
<td>Primitive</td>
<td>Required</td>
<td>Object for which you want to verify the type.</td>
</tr>
</tbody>
</table>
Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see N/util Module Script Sample.

```javascript
//Add additional code
...
var records = ['Sales Order', 'Invoice', 'Item Fulfillment'];
util.isArray(records); // returns true

var record = 'Sales Order';
util.isArray(record); // returns false
...
//Add additional code
```

**util.isBoolean(obj)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns true if the <code>obj</code> parameter is a boolean and false otherwise.</td>
<td>boolean true</td>
<td>All script types</td>
<td>None</td>
<td>N/util Module</td>
<td>2016.1</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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>obj</td>
<td>Object</td>
<td>Required</td>
<td>Object for which you want to verify the type.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see N/util Module Script Sample.

```javascript
//Add additional code
...
var flag = true;
util.isBoolean(flag); // returns true
util.Boolean(true);   // returns true
util.Boolean(1);      // returns false
...
//Add additional code
```
## util.isDate(obj)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns true if the <code>obj</code> parameter is a JavaScript Date object and false otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>`boolean true</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None                                                                              |
| Module             | N/util Module                                                                     |
| Since              | 2016.1                                                                            |

### 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><code>obj</code></td>
<td>`Object</td>
<td>Primitive`</td>
<td>Required</td>
<td>Object for which you want to verify the type.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
var todaysDate = new Date();
util.isDate(todaysDate);  // returns true
util.isDate(new Date());  // returns true
var today = "September 28, 2015";
util.isDate(today);       // returns false
...
//Add additional code
```

## util.isFunction(obj)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns true if the <code>obj</code> parameter is a JavaScript Function object and false otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>`boolean true</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None                                                                              |
| Module             | N/util Module                                                                     |
| Since              | 2016.1                                                                            |
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>obj</td>
<td>Object</td>
<td>Required</td>
<td>Object for which you want to verify the type.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see [N/util Module Script Sample](#).

```javascript
//Add additional code
...
function test() {}
var test2 = function() {};

util.isFunction(test);   // returns true
util.isFunction(test2);  // returns true
...
//Add additional code
```

`util.isNumber(obj)`

**Method Description**

Returns true if the `obj` parameter is a JavaScript Number object or primitive, and false otherwise.

**Returns**

boolean true | false

**Supported Script Types**

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

None

**Module**

N/util Module

**Since**

2016.1

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>obj</td>
<td>Object</td>
<td>Required</td>
<td>Object for which you want to verify the type.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

Syntax

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see [N/util Module Script Sample](#).

```javascript
//Add additional code
...
util.isNumber(112);                // returns true
util.isNumber("112");             // returns false
```
util.isNumber(NaN);                // returns true

var testNum = 112;
util.isNumber(testNum.valueOf());  // returns true
...
//Add additional code

util.isObject(obj)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns true if the obj parameter is a plain JavaScript object(new Object()) or () for example, and false otherwise. Use this method, for example, to verify that a variable is a JavaScript object and not a JavaScript Function.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>boolean true</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/util Module</td>
</tr>
<tr>
<td>Since</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

**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>obj</td>
<td>Object</td>
<td>Primitive</td>
<td>Required</td>
<td>Object for which you want to verify the type.</td>
</tr>
</tbody>
</table>

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see N/util Module Script Sample.

```javascript
//Add additional code
...
util.isObject({});                // returns true
util.isObject(function() {});     // returns false
...
//Add additional code
```

util.isRegExp(obj)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns true if the obj parameter is a JavaScript RegExp object, and false otherwise.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>boolean true</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>
## 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>obj</td>
<td>Object</td>
<td>Primitive</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Object for which you want to verify the type.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see N/util Module Script Sample.

```javascript
// Add additional code
...
util.isRegExp(/this is a regexp/); // returns true
util.isRegExp(new RegExp('this is another regexp')); // returns true
...
// Add additional code
```

### util.isString(obj)

**Method Description**

Returns true if the `obj` parameter is a JavaScript String object or primitive, and false otherwise.

**Returns**

boolean true | false

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/util Module

**Since**

2016.1

## 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>obj</td>
<td>Object</td>
<td>Primitive</td>
<td>Required</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Object for which you want to verify the type.</td>
<td>2016.1</td>
</tr>
</tbody>
</table>

## Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see N/util Module Script Sample.

```javascript
// Add additional code
...
util.isString(obj)
...
// Add additional code
```
util.isString('');                            // returns true
util.isString('a string');                    // returns true
var myString = new String('another string');
util.isString(myString);                      // returns true
util.isString(null);                          // returns false
...                                          //Add additional code

**util.nanoTime()**

<table>
<thead>
<tr>
<th><strong>Method Description</strong></th>
<th>Returns the current time (epoch) in nanoseconds. You can use this method to measure elapsed time between two events.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td>string</td>
</tr>
</tbody>
</table>
| **Supported Script Types** | Server-side scripts  
For more information, see the help topic *SuiteScript 2.0 Script Types.* |
| **Governance**         | None                                                                                                       |
| **Module**             | N/util Module                                                                                               |
| **Since**              | 2016.1                                                                                                      |

**Syntax**

⚠️ **Important:** The following code snippet shows the syntax for this member. It is not a functional example. It demonstrates how to calculate the number of nanoseconds between two calls to *util.nanoTime()* For a full script sample, see *N/util Module Script Sample.*

```javascript
//Add additional code
var startTime = util.nanoTime();
...
var elapsedTime = util.nanoTime() - startTime;
...
//Add additional code
```

**util.each(iterable, callback)**

| **Method Description** | Iterates over each member in an Object or Array.  
This method calls the *callback* function on each member of the *iterable.* |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Returns</strong></td>
<td>The original collection as an Object</td>
</tr>
</tbody>
</table>
| **Supported Script Types** | All script types  
For more information, see the help topic *SuiteScript 2.0 Script Types.* |
| **Governance**         | None                                                                 |
| **Module**             | N/util Module                                                                 |
### util.each(iterable, callback)

**Method Description**
Method used to copy the properties in a source object to a destination object. Returns the destination object.

You can use this method to merge two objects.

**Returns**
The Object receiving the properties copied from the contributor

**Supported Script Types**
All script types

**Governance**
None

**Module**
N/util Module

**Since**
2016.1

#### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a full script sample, see N/util Module Script Sample.

```plaintext
//Add additional code
...
// Iterate through the object and set the key-value pairs on the record
util.each(itemList, function(quantity, itemId){
    rec.selectNewLine('item');
    rec.setCurrentSublistValue('item','item',itemId);
    rec.setCurrentSublistValue('item','quantity',quantity);
    rec.commitLine('item');
});
...
//Add additional code
```

### util.extend(receiver, contributor)

**Method Description**
Method used to copy the properties in a source object to a destination object. Returns the destination object.

You can use this method to merge two objects.

**Returns**
The Object receiving the properties copied from the contributor

**Supported Script Types**
All script types

**Governance**
None

**Module**
N/util Module

**Since**
2016.1

#### Syntax

**Important:** The following code snippets shows the syntax for this member. It is not a functional example. For a full script sample, see N/util Module Script Sample.

This snippet shows combining two objects without the same keys:

```plaintext
//Add additional code
```
The following snippet shows overriding two objects with a few similar keys:

```javascript
// Add additional code
...
var colors = {};
var firstSet = {'color1':'red',
  'color2':'yellow',
  'color3':'blue'};
var secondSet = {'color4':'green',
  'color5':'orange',
  'color6':'violet'};

// Extends colors object with the information in firstSet
// Colors will get {'color1':'red','color2':'yellow','color3':'blue'}
util.extend(colors, firstSet);

// Extends colors object with the information in secondSet and overrides the value if there are similar keys
// Colors will get
// {'color1':'red','color2':'green','color3':'orange','color4':'violet'}
util.extend(colors, secondSet);
});
...  
// Add additional code
```

### N/workflow Module

This module loads the workflow module to initiate new workflow instances or trigger existing workflow instances.
N/workflow Module Members

N/workflow Module Script Sample

N/workflow Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>workflow.##initiate##[number]</td>
<td>Server-side scripts</td>
<td>Initiates a workflow on-demand. This method is the programmatic equivalent of the Initiate Workflow Action action in SuiteFlow. Returns the internal ID (number) of the workflow instance used to track the workflow against the record.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>workflow.##trigger##[number]</td>
<td>Server-side scripts</td>
<td>Triggers a workflow on a record. The actions and transitions of the workflow are evaluated for the record in the workflow instance, based on the current state for the workflow instance. Returns the internal ID (number) of the workflow instance used to track the workflow against the record.</td>
<td></td>
</tr>
</tbody>
</table>

N/workflow Module Script Sample

The following example searches for a specific workflow deployed on the customer record and then executes it.

This sample script uses the require function so that you can copy it into the debugger and test it. Keep in mind that you must use the define function in your entry point script (the script you attach to a script record). For additional information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

```javascript
/**
* @NApiVersion 2.x
*/
require(['N/workflow', 'N/search', 'N/error', 'N/record'],
  function(workflow, search, error, record) {
    function initiateWorkflow() {
      var workflowInstanceId = workflow.initiate({
        recordType: 'customer',
        recordId: 24,
        workflowId: 'customworkflow_myWorkFlow'
      });
      var customerRecord = record.load({
        type: record.Type.CUSTOMER,
        id: 24
      });
    }
  });
```
workflow.initiate(options)

### Method Description
Initiates a workflow on-demand. This method is the programmatic equivalent of the Initiate Workflow Action action in SuiteFlow.

Returns the internal ID of the workflow instance used to track the workflow against the record.

### Returns
number

### Supported Script Types
All server-side scripts

For more information, see the help topic SuiteScript 2.0 Script Types

### Governance
20 usage units

### Module
N/workflow Module

### Since
2015.2

### 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>options.recordType</td>
<td>number</td>
<td>required</td>
<td>The record type ID of the workflow base record. For example, use 'customer', 'salesorder', or 'lead'. This is the Record Type field on the Workflow Definition Page.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.recordId</td>
<td>string</td>
<td>number</td>
<td>required</td>
<td>The internal ID of the base record</td>
</tr>
<tr>
<td>options.workflowId</td>
<td>string</td>
<td>number</td>
<td>required</td>
<td>The internal ID (number) or script ID (string) for the workflow definition. This is the ID field on the Workflow Definition Page.</td>
</tr>
<tr>
<td>options.defaultValues</td>
<td>Object</td>
<td>optional</td>
<td>The object that contains key/value pairs to set default values on fields specific to the workflow. These can include fields on the Workflow Definition Page or workflow and state Workflow Custom Fields.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/workflow Module Script Sample.

```javascript
//Add additional code
```

//Add additional code
```javascript
var workflowInstanceId = workflow.initiate({
  recordType: 'customer',
  recordId: 24,
  workflowId: 'customworkflow_myWorkflow'
});
...

// Add additional code

workflow.trigger(options)
```

### workflow.trigger(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Triggers a workflow on a record. The actions and transitions of the workflow are evaluated for the record in the workflow instance, based on the current state for the workflow instance. Returns the internal ID of the workflow instance used to track the workflow against the record.</th>
</tr>
</thead>
</table>
| Supported Script Types | All server-side scripts  
For more information, see the help topic SuiteScript 2.0 Script Types |
| Governance | 20 usage units |
| Module | N/workflow Module |
| Since | 2015.2 |

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.recordType</td>
<td>number</td>
<td>required</td>
<td>The record type ID of the workflow base record. For example, use 'customer', 'salesorder', or 'lead'. This is the Record Type field on the Workflow Definition Page.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.recordId</td>
<td>string</td>
<td>required</td>
<td>The internal ID of the base record</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.workflowId</td>
<td>string</td>
<td>required</td>
<td>The internal ID (number) or script ID (string) for the workflow definition. This is the ID field on the Workflow Definition Page.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.workflowInstanceId</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of the workflow instance.</td>
<td>2015.2</td>
</tr>
<tr>
<td>options.actionId</td>
<td>string</td>
<td>optional</td>
<td>The internal ID of a button that appears on the record in the workflow. Use this parameter to trigger the workflow as if the specified button were clicked.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### N/workflow Module

#### 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>options.stateId</td>
<td>string</td>
<td>optional</td>
<td>The internal ID (number) or script ID (string) of the workflow instance.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

**Important:** The following code snippet shows the syntax for this member. It is not a functional example. For a complete script example, see N/workflow Module Script Sample.

```javascript
//Add additional code
...
var workflowInstanceId = workflow.trigger({
  recordType: 'salesorder',
  recordId: 1234,
  workflowId: 'custworkflow_name',
  defaultValues: p
  actionId: workflowaction25
});
...
//Add additional code
```

### N/xml Module

Load the xml module to validate, parse, read, and modify XML documents.

- **N/xml Module Members**
- **Parser Object Members**
- **XPath Object Members**
- **Node Object Members**
- **Document Object Members**
- **Element Object Members**
- **Attr Object Members**
- **N/xml Module Script Samples**

#### N/xml Module Members

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td>xmlParser</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the functionality used by NetSuite to parse XML.</td>
</tr>
<tr>
<td></td>
<td>xmlXPath</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Encapsulates the functionality used by NetSuite to run XPath expressions. XPath is a standard for enumerating paths in an XML document collection.</td>
</tr>
<tr>
<td></td>
<td>xmlNode</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Represents a generic XML node in an XML document. A node can be a Document, Element, or Attribute.</td>
</tr>
</tbody>
</table>
### xml.Document

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
</table>

### xml.Element

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xml.Element</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Represents an element in an XML document. Elements may contain attributes, other elements, or text. If an element contains text, the text is represented in a text node of type TEXT_NODE.</td>
</tr>
</tbody>
</table>

### xml.Attr

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xml.Attr</td>
<td>Object</td>
<td>Client and server-side scripts</td>
<td>Represents an attribute node of an xml.Element object.</td>
</tr>
</tbody>
</table>

### xml.escape(options)

<table>
<thead>
<tr>
<th>Method</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xml.escape(options)</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Prepares a string for use in XML by escaping XML markup, such as angle brackets, quotation marks, and ampersands.</td>
</tr>
</tbody>
</table>

### xml.â##validateâ##(options)

<table>
<thead>
<tr>
<th>Method</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
</table>

### Enum

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>xml.NodeType</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Enumeration that holds the string values for the supported node types. The Node.nodeType property is defined by one of the values in this enum.</td>
</tr>
</tbody>
</table>

### Parser Object Members

The following members are called on the xml.Parser object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>xml.â##fromStringâ##(options)</td>
<td>xml.Document</td>
<td>Client and server-side scripts</td>
<td>Parses a string into a W3C XML document object.</td>
</tr>
<tr>
<td>Method</td>
<td>xml.â##toStringâ##(options)</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Converts (serializes) an xml.Document object into a string.</td>
</tr>
</tbody>
</table>

### XPath Object Members

The following members are called on the xml.XPath object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>XPath.select(options)</td>
<td>xml.Node[]</td>
<td>Client and server-side scripts</td>
<td>Selects an array of nodes from an XML document using an XPath expression.</td>
</tr>
</tbody>
</table>
# Node Object Members

The following members are called on the `xml.Node` object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td><code>appendChild</code></td>
<td><code>xml.Node</code></td>
<td>Client and server-side scripts</td>
<td>Appends a node after the last child node of a specific element node. Returns the new child node.</td>
</tr>
<tr>
<td></td>
<td><code>cloneNode</code></td>
<td><code>xml.Node</code></td>
<td>Client and server-side scripts</td>
<td>Creates a copy of a node. Returns the copied node.</td>
</tr>
<tr>
<td></td>
<td><code>compareDocumentPosition</code></td>
<td><code>number</code></td>
<td>Client and server-side scripts</td>
<td>Returns a number that reflects where two nodes are located, compared to each other.</td>
</tr>
<tr>
<td></td>
<td><code>hasAttributes</code></td>
<td><code>boolean</code></td>
<td>Client and server-side scripts</td>
<td>Returns <code>true</code> if the current node has child nodes or returns <code>false</code> if the current node does not have child nodes.</td>
</tr>
<tr>
<td></td>
<td><code>hasChildNodes</code></td>
<td><code>boolean</code></td>
<td>Client and server-side scripts</td>
<td>Returns <code>true</code> if the current node has any attributes. Note that only element nodes can have attributes.</td>
</tr>
<tr>
<td></td>
<td><code>insertBefore</code></td>
<td><code>xml.Node</code></td>
<td>Client and server-side scripts</td>
<td>Inserts a new child node before an existing child node for the current node.</td>
</tr>
<tr>
<td></td>
<td><code>isDefaultNamespace</code></td>
<td><code>boolean</code></td>
<td>Client and server-side scripts</td>
<td>Returns <code>true</code> if the specified namespace uniform resource identifier (URI) is the default namespace for the current node or returns <code>false</code> if the specified namespace is not the default namespace.</td>
</tr>
<tr>
<td></td>
<td><code>isEqualNode</code></td>
<td><code>boolean</code></td>
<td>Client and server-side scripts</td>
<td>Returns <code>true</code> if two nodes are equal or returns <code>false</code> if two nodes are not equal.</td>
</tr>
<tr>
<td></td>
<td><code>isSameNode</code></td>
<td><code>boolean</code></td>
<td>Client and server-side scripts</td>
<td>Returns <code>true</code> if two nodes reference the same object or returns <code>false</code> if two nodes do not reference the same object.</td>
</tr>
<tr>
<td></td>
<td><code>lookupNamespaceURI</code></td>
<td><code>string</code></td>
<td>Client and server-side scripts</td>
<td>Returns the namespace uniform resource identifier (URI) that matches the specified namespace prefix.</td>
</tr>
<tr>
<td></td>
<td><code>lookupPrefix</code></td>
<td><code>string</code></td>
<td>Client and server-side scripts</td>
<td>Returns the namespace prefix associated with the specified namespace uniform resource identifier (URI).</td>
</tr>
<tr>
<td></td>
<td><code>normalize</code></td>
<td><code>void</code></td>
<td>Client and server-side scripts</td>
<td>Puts all text nodes underneath a node, including attribute nodes, into a normal form.</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Node removesChild(options)</td>
<td>Node</td>
<td>xml.Node</td>
<td>Client and server-side scripts</td>
<td>Removes the specified child node. Returns the removed child node.</td>
</tr>
<tr>
<td>Node replaceChild(ol)</td>
<td>Node</td>
<td>xml.Node</td>
<td>Client and server-side scripts</td>
<td>Replaces a specific child node with another child node in a list of child nodes.</td>
</tr>
</tbody>
</table>

**Property**

<table>
<thead>
<tr>
<th>Property</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node attributes</td>
<td>Object (read-only)</td>
<td>Client and server-side scripts</td>
<td>Key-value pairs for all attributes for an xml.Element node. Returns null for all other node types.</td>
<td></td>
</tr>
<tr>
<td>Node.baseURI</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Absolute base uniform resource identifier (URI) of a node or null if the URI cannot be determined.</td>
<td></td>
</tr>
<tr>
<td>Node.childNodes</td>
<td>xml.Node[] (read-only)</td>
<td>Client and server-side scripts</td>
<td>Array of all child nodes of a node or an empty array if there are no child nodes.</td>
<td></td>
</tr>
<tr>
<td>Node.firstChild</td>
<td>xml.Node (read-only)</td>
<td>Client and server-side scripts</td>
<td>First child node for a specific node or null if there are no child nodes.</td>
<td></td>
</tr>
<tr>
<td>Node.lastChild</td>
<td>xml.Node (read-only)</td>
<td>Client and server-side scripts</td>
<td>Last child node for a specific node or null if there is no last child node.</td>
<td></td>
</tr>
<tr>
<td>Node.localName</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>The local part of the qualified name of a node.</td>
<td></td>
</tr>
<tr>
<td>Node.namespaceURI</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>The namespace uniform resource identifier (URI) of a node or null if there is no namespace URI for the node.</td>
<td></td>
</tr>
<tr>
<td>Node.nextSibling</td>
<td>xml.Node (read-only)</td>
<td>Client and server-side scripts</td>
<td>The next node in a node list or null if the current node is the last node.</td>
<td></td>
</tr>
<tr>
<td>Node.nodeName</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>Name of a node, depending on the type. For example, for a node of type xml.Element, the name is the name of the element.</td>
<td></td>
</tr>
<tr>
<td>Node.nodeType</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The type of node defined as a value from the xml.NodeType enum.</td>
<td></td>
</tr>
<tr>
<td>Node.nodeValue</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The value of a node, depending on its type.</td>
<td></td>
</tr>
<tr>
<td>Node.parentNode</td>
<td>xml.Node (read-only)</td>
<td>Client and server-side scripts</td>
<td>The parent node of a node.</td>
<td></td>
</tr>
<tr>
<td>Node.prefix</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>The namespace prefix of the node, or null if the node does not have a namespace.</td>
<td></td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
<td>Description</td>
</tr>
<tr>
<td>-------------</td>
<td>---------------------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Node.previousSibling</td>
<td>xml.Node (read-only)</td>
<td>Client and server-side</td>
<td>The previous node in a node list or null if the current node is the first node.</td>
</tr>
<tr>
<td></td>
<td>Node.textContent</td>
<td>string</td>
<td>Client and server-side</td>
<td>The textual content of a node and its descendants.</td>
</tr>
</tbody>
</table>

**Document Object Members**

> **Note:** In addition to the Document object members, Document objects inherit the members of the Node object. The methods and properties associated with a Node object can be used as members of a Document object. For more information, see [Node Object Members](#).

The following members are called on the `xml.Document` object.

<table>
<thead>
<tr>
<th>Method</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document.adoptNode(options)</td>
<td>xml.Node</td>
<td>Client and server-side</td>
<td>Attempts to adopt a node from another document to this document.</td>
</tr>
<tr>
<td>Document.createAttribute(options)</td>
<td>xml.Attr</td>
<td>Client and server-side</td>
<td>Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value.</td>
</tr>
<tr>
<td>Document.createAttributeNS(options)</td>
<td>xml.Attr</td>
<td>Client and server-side</td>
<td>Creates an attribute node of type ATTRIBUTE_NODE, with the specified namespace value and optional specified value.</td>
</tr>
<tr>
<td>Document.createElement(options)</td>
<td>xml.Element</td>
<td>Client and server-side</td>
<td>Creates a new node of type ELEMENT_NODE with the specified name.</td>
</tr>
<tr>
<td>Document.createElementNS(options)</td>
<td>xml.Element</td>
<td>Client and server-side</td>
<td>Creates a new node of type ELEMENT_NODE with the specified namespace URI and name.</td>
</tr>
<tr>
<td>Document.createProcessingInstruction(options)</td>
<td>xml.Node</td>
<td>Client and server-side</td>
<td>Creates a new node of type PROCESSING_INSTRUCTION_NODE with the specified target and data.</td>
</tr>
<tr>
<td>Document.createTextNode(options)</td>
<td>xml.Node</td>
<td>Client and server-side</td>
<td>Creates a new node of type TEXT_NODE.</td>
</tr>
<tr>
<td>Document.getElementById(options)</td>
<td>xml.Element</td>
<td>Client and server-side</td>
<td>Returns the element that has an ID attribute with the specified value as an xml.Element object.</td>
</tr>
<tr>
<td>Document.getElementsByTagName(options)</td>
<td>xml.Element[]</td>
<td>Client and server-side</td>
<td>Returns an array of xml.Element objects with a specific tag name, in...</td>
</tr>
<tr>
<td>Member Type</td>
<td>Name</td>
<td>Return Type / Value Type</td>
<td>Supported Script Types</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
</tr>
<tr>
<td>Property</td>
<td>Document.doctype</td>
<td>Object (read-only)</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Property</td>
<td>Document.documentURI</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Property</td>
<td>Document.inputEncoding</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Property</td>
<td>Document.xmlEncoding</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td>Property</td>
<td>Document.xmlStandalone</td>
<td>boolean true</td>
<td>false</td>
</tr>
<tr>
<td>Property</td>
<td>Document.xmlVersion</td>
<td>string</td>
<td>Client and server-side scripts</td>
</tr>
</tbody>
</table>

Element Object Members

**Note:** In addition to the Element object members, Element objects inherit the members of the Node object. The methods and properties associated with a Node object can be used as members of a Element object. For more information, see Node Object Members.

The following members are called on the xml.Element object.

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>Element.getAttribute</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Returns the value of the specified attribute.</td>
</tr>
<tr>
<td>Method</td>
<td>Element.getAttributeNode</td>
<td>xml.Attr</td>
<td>Client and server-side scripts</td>
<td>Retrieves an attribute node by name.</td>
</tr>
<tr>
<td>Method</td>
<td>Element.getAttributeNodeNS</td>
<td>xml.Attr</td>
<td>Client and server-side scripts</td>
<td>Returns an attribute node with the specified namespace URI and local name.</td>
</tr>
<tr>
<td>Method</td>
<td>Element.getAttributeNS</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Returns an attribute value with the specified namespace URI and local name.</td>
</tr>
</tbody>
</table>
### Element Methods

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element.prototypeElementsByTag</td>
<td>options</td>
<td>xml.Element[]</td>
<td>Client and server-side scripts</td>
<td>Returns an array of descendant xml.Element objects with a specific tag name, in the order in which they appear in the XML document.</td>
</tr>
<tr>
<td>Element.prototypeElementsByTagNS</td>
<td>options</td>
<td>xml.Element[]</td>
<td>Client and server-side scripts</td>
<td>Returns an array of descendant xml.Element objects with a specific tag name and namespace, in the order in which they appear in the XML document.</td>
</tr>
<tr>
<td>Element.prototypehasAttribute</td>
<td>options</td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the current element has an attribute with the specified name or if that attribute has a default value. Otherwise, returns false.</td>
</tr>
<tr>
<td>Element.prototypehasAttributeNS</td>
<td>options</td>
<td>boolean true</td>
<td>Client and server-side scripts</td>
<td>Returns true if the current element has an attribute with the specified local name and namespace or if that attribute has a default value. Otherwise, returns false.</td>
</tr>
<tr>
<td>Element.prototyperemoveAttribute</td>
<td>options</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Removes the attribute with the specified name.</td>
</tr>
<tr>
<td>Element.prototyperemoveAttributeNode</td>
<td>options</td>
<td>xml.Attr</td>
<td>Client and server-side scripts</td>
<td>Removes the attribute specified as a xml.Attr object.</td>
</tr>
<tr>
<td>Element.prototyperemoveAttributeNS</td>
<td>options</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Removes the attribute with the specified namespace URI and local name.</td>
</tr>
<tr>
<td>Element.prototypesetAttribute</td>
<td>options</td>
<td>void</td>
<td>Client and server-side scripts</td>
<td>Adds a new attribute with the specified name. If an attribute with that name is already present in the element, its value is changed to the value specified in method argument.</td>
</tr>
<tr>
<td>Element.prototypesetAttributeNode</td>
<td>options</td>
<td>xml.Attr</td>
<td>Client and server-side scripts</td>
<td>Adds the specified attribute node. If an attribute with the same name is already present in the element, it is replaced by the new one.</td>
</tr>
<tr>
<td>Element.prototypesetAttributeNodeNS</td>
<td>options</td>
<td>xml.Attr</td>
<td>Client and server-side scripts</td>
<td>Adds the specified attribute node. If an attribute with the same local name and namespace URI is already present in the element, it is replaced by the new one.</td>
</tr>
</tbody>
</table>

### Element Property

| Property | Element.tagName | String (read-only) | Client and server-side scripts | The tag name of this xml.Element object. |

### Attr Object Members

The following members are called on the xml.Attr object.
### N/xml Module

<table>
<thead>
<tr>
<th>Member Type</th>
<th>Name</th>
<th>Return Type / Value Type</th>
<th>Supported Script Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property</td>
<td>Attr.name</td>
<td>string (read-only)</td>
<td>Client and server-side scripts</td>
<td>The name of an attribute.</td>
</tr>
<tr>
<td></td>
<td>Attr.ownerElement</td>
<td>xml.Element (read-only)</td>
<td>Client and server-side scripts</td>
<td>The xml.Element object that is the parent of the xml.Attr object.</td>
</tr>
<tr>
<td></td>
<td>Attrspecified</td>
<td>boolean true</td>
<td>false</td>
<td>Client and server-side scripts</td>
</tr>
<tr>
<td></td>
<td>Attr.value</td>
<td>string</td>
<td>Client and server-side scripts</td>
<td>Value of an attribute. The value of the attribute is returned as a string. Character and general entity references are replaced with their values.</td>
</tr>
</tbody>
</table>

### N/xml Module Script Samples

These samples use the `require` function, so that you can copy each script into the debugger and test it. Keep in mind that you must use the define function in your entry point script (the script you attach to a script record). For more information, see the help topics SuiteScript 2.0 Script Basics and SuiteScript 2.0 Script Types.

For help with writing scripts in SuiteScript 2.0, see the help topics SuiteScript 2.0 Hello World and SuiteScript 2.0 Entry Point Script Creation and Deployment.

The N/xml module sample references the following XML file, BookSample.xml:

```xml
  <b:book category="cooking">
    <b:title lang="en">Everyday Italian</b:title>
    <b:author>Giada De Laurentiis</b:author>
    <b:year>2005</b:year>
    <b:price>30.00</b:price>
  </b:book>
  <b:book category="children">
    <b:title lang="en">Harry Potter</b:title>
    <b:author>J K. Rowling</b:author>
    <b:year>2005</b:year>
    <b:price>29.99</b:price>
  </b:book>
  <b:book category="web">
    <b:title lang="en">XQuery Kick Start</b:title>
    <b:author>James McGovern</b:author>
    <b:author>Per Bothner</b:author>
    <b:author>Kurt Cagle</b:author>
    <b:author>James Linn</b:author>
    <b:author>Vaidyanathan Nagarajan</b:author>
    <b:year>2003</b:year>
    <b:price>49.99</b:price>
  </b:book>
  <b:book category="web" cover="paperback">
    <b:title lang="en">Learning XML</b:title>
    <b:author>Erik T. Ray</b:author>
  </b:book>
</bookstore>
```
The following Suitelet example loads an XML file from the file cabinet, iterates through the individual book nodes, and accesses the child node values using two common methods: (through firstChild/nextSibling/etc and through getElementsByTagName)

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
require([ 'N/xml', 'N/file' ],
    function(xml, file) {
        return {
            onRequest : function(options) {
                var sentence = ''; // Initialize the sentence variable
                var xmlFileContent = file.load('SuiteScripts/BookSample.xml').getContents(); // Load the XML file
                var xmlDocument = xml.Parser.fromString({
                    text : xmlFileContent
                }); // Parse the XML string
                var bookNode = xml.XPath.select({
                    node : xmlDocument,
                    xpath : '//book'
                }); // Select all book nodes in the XML document
                for (var i = 0; i < bookNode.length; i++) {
                    var title = bookNode[i].firstChild.nextSibling.textContent; // Access the title
                    var author = bookNode[i].getElementsByTagName({
                        tagName : 'b:author'
                    })[0].textContent; // Access the author
                    sentence += 'Author: ' + author + ' wrote ' + title + '.
'; // Add the sentence to the result
                }
                options.response.write(sentence); // Write the result to the response
            }
        }
    });
```

The following output is produced from the sample code when used with the BookSample.xml document:

```
Author: Giada De Laurentiis wrote Everyday Italian.
Author: J K. Rowling wrote Harry Potter.
Author: James McGovern wrote XQuery Kick Start.
Author: Erik T. Ray wrote Learning XML.
```

In the following example, the XML parser parses the XML string stored in the xmlString variable. Then, the script selects all `config` elements in the `xmlDocument` node, loops through them and logs their content.

```javascript
/**
 * @NApiVersion 2.x
 * @NScriptType Suitelet
 */
```
require(['N/xml'], function(xml) {
    var xmlString = '<?xml version="1.0" encoding="UTF-8"?>
<config date="1465467658668" transient="false">Some content</config> ;

    var xmlDocument = xml.Parser.fromString({
        text : xmlString
    });

    var bookNode = xml.XPath.select({
        node : xmlDocument,
        xpath : '//config'
    });

    var i;
    for (i = 0; i < bookNode.length; i++) {
        log.debug('Config content', bookNode[i].textContent);
    }
});

The following example modifies an XML file.

require(['N/xml'], function(xml){
    var bookShelf = xml.Parser.fromString(file.load('SuiteScripts/books.xml').getContents());

    var newBookNode = xmlData.createElement("book");
    var newTitleNode = xmlData.createElement("title");
    var newTitleNodeValue = xmlData.createTextNode("'");
    var newAuthorNode = xmlData.createElement("author");
    var newAuthorNodeValue = xmlData.createTextNode("'");
    newTitleNode.appendChild(newTitleNodeValue);
    newAuthorNode.appendChild(newAuthorNodeValue);
    newBookNode.appendChild(newTitleNode);
    newBookNode.appendChild(newAuthorNode);

    var newbook = bookShelf.appendChild({
        newChild : newBookNode
    });
});

xml.Parser

<table>
<thead>
<tr>
<th>Object Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encapsulates the functionality used by NetSuite to parse an XML document.</td>
</tr>
<tr>
<td>For a complete list of this object's methods, see Parser Object Members.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>All script types</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/xml Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

//Add additional code
... var parserObj = xml.Parser;
...
//Add additional code

Parser.fromString(options)

Method Description | Parsers a String into a W3C XML document object. This API is useful if you want to navigate/query a structured XML document more effectively using either the Document API or NetSuite built-in XPath functions.

| Note: You can also use this method to validate your XML. If you pass a malformed string in as the options.text argument, Parser.fromString returns an SSS_XML_DOM_EXCEPTION error.

Returns | xml.Document

Supported Script Types | All script types
Governance | None
Module | N/xml Module
Since | 2015.2

Parameters

| Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.text | string | Required | String being converted to an xml.Document.

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
</table>
| SSS_XML_DOM_EXCEPTION | The input XML string is malformed.

Syntax

```javascript
//Add additional code
...
var xmlDocument = xml.Parser.fromString({
  text : xmlStringContent
});
...
//Add additional code
```
**Parser.toString(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Converts (serializes) an <code>xml.Document</code> object into a string. This API is useful, for example, if you want to serialize and store an <code>xml.Document</code> in a custom field.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td><code>N/xml Module</code></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

ℹ️ **Note:** The `options` parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>

**Syntax**

```javascript
//Add additional code
...
var xmlStringContent = xml.Parser.toString({
    document : xmlDocument
});
...
//Add additional code
```

**xml.XPath**

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Encapsulates the functionality to run XPath expressions. For a complete list of this object's methods, see <a href="#">XPath Object Members</a>.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Module</td>
<td><code>N/xml Module</code></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var xpath = xml.XPath;
...
//Add additional code
```
XPath.select(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Selects an array of nodes from an XML that match an XPath expression.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>xml.Node[]</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.node</td>
<td>xml.Node</td>
<td>Required</td>
<td>XML node being queried.</td>
</tr>
<tr>
<td>options.xpath</td>
<td>string</td>
<td>Required</td>
<td>XPath expression used to query node.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var bookNode = xml.XPath.select({
   node : xmlDocument,
   xpath : '//book'
});
...
//Add additional code
```

xml.Node

Object Description

Represents a single node in an XML document tree. The XML DOM presents a document as a hierarchy of node objects. See the xml.NodeType enum for a list of possible node types.

You can use this object to work with a child node, or nested nodes.

NetSuite supports a subset of W3C DOM methods. For a complete list of this object’s methods and properties, see Node Object Members.

For other code snippets that use this object, see the syntax sample that follows, as well as Node.childNodes and N/xml Module Script Samples.

Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Module

N/xml Module

Since

2015.2
Syntax

```javascript
//Add additional code
...
var bookNode = xml.XPath.select({
    node : xmlDocument,
    xpath : '//book'
});
...
//Add additional code
```

### Node.appendChild(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Appends a node after the last child node of a specific element node. Returns the new child node.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>xml.Node</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types                               For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.</td>
<td>Node cannot be appended.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var bookShelf = xml.Parser.fromString(file.load('SuiteScripts/books.xml').getContents());
var newBookNode = xmlData.createElement("book");
var newTitleNode = xmlData.createElement("title");
var newTitleNodeValue = xmlData.createTextNode('');
var newAuthorNode = xmlData.createElement("author");
var newAuthorNodeValue = xmlData.createTextNode('');
```
newTitleNode.appendChild(newTitleNodeValue);
newAuthorNode.appendChild(newAuthorNodeValue);
newBookNode.appendChild(newTitleNode);
newBookNode.appendChild(newAuthorNode);

var newbook = bookShelf.appendChild({
    newChild : newBookNode
});
...
//Add additional code

Node.cloneNode(options)

Method Description: Creates a copy of a node. Returns the copied node.

Returns: xml.Node

Supported Script Types: All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance: None

Module: N/xml Module

Since: 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.deep</td>
<td>boolean</td>
<td>true</td>
<td>Optional</td>
</tr>
<tr>
<td></td>
<td>true</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

Syntax

//Add additional code
...
var copiednode = elem[0].cloneNode({
    deep : true
});
...
//Add additional code

Node.compareDocumentPosition(options)

Method Description: Returns a number that reflects where two nodes are located, compared to each other. Returns one of the following numbers:

- 1. The two nodes do not belong to the same document.
- 2. The specified node comes before the current node.
- 4. The specified node comes after the current node.
- 8. The specified node contains the current node.
- 16. The current node contains the specified node.
- 32. The specified and current nodes do not have a common container node or the two nodes are different attributes of the same node.

**Note:** The return value can be a combination of the above values. For example, a return value of 20 means the specified node is contained by the current node, a value of 16, and the specified node follows the current node, a value of 4.

**Important:** This method is not supported on Internet Explorer.

### Returns
- **number**

### Supported Script Types
- **All script types**
- For more information, see the help topic [SuiteScript 2.0 Script Types](#).

### Governance
- **None**

### Module
- **N/xml Module**

### Since
- **2015.2**

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.other</td>
<td>xml.Node</td>
<td>Required</td>
<td>The node to compare with the current node.</td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected xml.Node or subclass: other</td>
<td>The options.other is of type xml.Node.</td>
</tr>
</tbody>
</table>

#### Syntax

```javascript
//Add additional code
...
var posCode = elem[0].compareDocumentPosition({
   other : parentNode[0]
});
...
//Add additional code
```

### Node.hasAttributes()

**Method Description**

Returns `true` if the current node has attributes defined, or `false` otherwise.
### Important:
This method is not supported on Internet Explorer.

<table>
<thead>
<tr>
<th>Returns</th>
<th>boolean true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

#### Syntax

```javascript
//Add additional code
...
var hasAttributes = parentNode[0].hasAttributes()
...
//Add additional code
```

### Node.hasChildNodes()

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns true if the current node has child nodes or returns false if the current node does not have child nodes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>boolean true</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Syntax

```javascript
//Add additional code
...
var hasChildren = parentNode[0].hasChildNodes()
...
//Add additional code
```

### Node.insertBefore(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Inserts a new child node before an existing child node for the current node. If the new child node is already in the list of children, this method removes the new child node and inserts it again.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>xml.Node</td>
</tr>
</tbody>
</table>

---

**SuiteScript 2.0 API Reference**

[Oracle NetSuite](https://www.oracle.com/netSuite/)

Supported Script Types | All script types  
| For more information, see the help topic SuiteScript 2.0 Script Types.

Governance | None

Module | N/xml Module

Since | 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.newChild</td>
<td>xml.Node</td>
<td>Required</td>
<td>The new child node to insert.</td>
</tr>
<tr>
<td>options.refChild</td>
<td>xml.Node</td>
<td>Required</td>
<td>The node before which to insert the new child node. If refChild is , the method inserts the new node at the end of the list of children.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.</td>
<td>Node cannot be inserted.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var insertednode = parentNode[0].insertBefore({
  newChild : elemlist1[0],
  refChild : elemlist2[0]
});
...
//Add additional code
```

Node.isDefaultNamespace(options)

**Method Description**

Returns `true` if the specified namespace uniform resource identifier (URI) is the default namespace for the current node or returns `false` if the specified namespace is not the default namespace.

See also `Node.namespaceURI`.

**Important:** This method is not supported on Internet Explorer.
Node.isEqualNode(options)

Method Description
Returns `true` if two nodes are equal or returns `false` if two nodes are not equal.
The two nodes are equal if they meet the following conditions:
- Both nodes have the same type.
- Both nodes have the same attributes and attribute values. The order of the attributes is not considered.
- Both nodes have equal lists of child nodes and the child nodes appear in the same order.

Note: Two nodes may be equal, even if they are not the same. See `Node.isSameNode(options)`.

Important: This method is not supported on Internet Explorer.

Returns
`boolean`: `true` | `false`

Supported Script Types
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/xml Module

Since
2015.2
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.other</td>
<td>xml.Node</td>
<td>Required</td>
<td>The node to compare with the current node.</td>
</tr>
</tbody>
</table>

**Note:** The options parameter is a JavaScript object.

Syntax

```javascript
//Add additional code
...
var isEqual = elem[0].isEqualNode({
  other : node
});
...
//Add additional code
```

Node.isSameNode(options)

**Method Description**

Returns `true` if two nodes reference the same object or returns `false` if two nodes do not reference the same object.

If two nodes are the same, all attributes have the same values and you can use methods on the two nodes interchangeably.

**Note:** Two nodes that are the same are also equal. See Node.isEqualNode(options).

**Important:** This method is not supported on Internet Explorer or Firefox.

**Returns**

| boolean     | `true` | `false` |

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/xml Module

**Since**

2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.other</td>
<td>xml.Node</td>
<td>Required</td>
<td>The node to compare with the current node.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
```
var isSame = elem[0].isSameNode({
    other : node
});
...
//Add additional code

Node.lookupNamespaceURI(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns the namespace uniform resource identifier (URI) that matches the specified namespace prefix. Returns null if the specified prefix does not have an associated URI.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types  For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.prefix</td>
<td>string</td>
<td>Required</td>
<td>Namespace prefix associated with the namespace URI.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var uri = parentNode[0].lookupNamespaceURI({
    prefix : '*'
});
...
//Add additional code
```

Node.lookupPrefix(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns the namespace prefix associated with the specified namespace uniform resource identifier (URI). Returns null if the specified URI does not have an associated prefix. If more than one prefix is associated with the namespace prefix, the namespace returned by this method depends on the module implementation.</th>
</tr>
</thead>
</table>

Important: This method is not supported on Internet Explorer.
returns string

Supported Script Types All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance None

Module N/xml Module

Since 2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI associated the namespace prefix.</td>
</tr>
</tbody>
</table>

Syntax

```js
//Add additional code
...
var prefix = parentNode[0].lookupPrefix({
  namespaceURI : '*'
});
...
//Add additional code
```

Node.normalize()

Method Description Puts all text nodes underneath a node, including attribute nodes, into a normal form. In normal form, only structure (such as elements, comments, processing instructions, CDATA sections, and entity references) separates text nodes. After normalization, there are no adjacent or empty text nodes.

Use this method if you require a particular document tree structure and want to make sure that the XML DOM view of a document is identical when you save and reload it.

Returns void

Supported Script Types All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance None

Module N/xml Module

Since 2015.2

Syntax

```js
//Add additional code
```
Node.removeChild(options)

Method Description | Removes the specified child node.
--- | ---
Returns | xml.Node
Supported Script Types | All script types
Governance | None
Module | N/xml Module
Since | 2015.2
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
</table>
| options.oldChild | xml.Node | Required | Node to remove.

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
</table>
| SSS_XML_DOM_EXCEPTION | NOT_FOUND_ERR: An attempt is made to reference a node in a context where it does not exist. | Node cannot be removed.

Syntax

```javascript
//Add additional code
...
var removednode = parentNode[0].removeChild({
  oldChild : node
});
...
//Add additional code
```

Node.replaceChild(options)

Method Description | Replaces a specific child node with another child node in a list of child nodes. If the new child node to add already exists in the list of child nodes, the node is first removed.
--- | ---
Returns | xml.Node
Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.newChild</td>
<td>xml.Node</td>
<td>Required</td>
<td>New child node to add.</td>
</tr>
<tr>
<td>options.oldChild</td>
<td>xml.Node</td>
<td>Required</td>
<td>Child node to replaced with the new node.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>NOT_FOUND_ERR: An attempt is made to reference a node in a context where it does not exist.</td>
<td>Child node cannot be found.</td>
</tr>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>HIERARCHY_REQUEST_ERR: An attempt was made to insert a node where it is not permitted.</td>
<td>Child node cannot be replaced.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var replacednode = parentNode.replaceChild({
    newChild : elem[2],
    oldChild : elem[1]
});
...
//Add additional code
```

Node.attributes

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types  For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
Node.baseURI

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node.baseURI</td>
<td>Absolute base uniform resource identifier (URI) of a node or null if the URI cannot be determined. For client scripts, this property always returns null.</td>
</tr>
</tbody>
</table>

**Note:** The format of this value is browser-specific.

<table>
<thead>
<tr>
<th>Type</th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/xml Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var baseuri = parentNode[0].baseURI;  
...
//Add additional code
```

Node.childNodes

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node.childNodes</td>
<td>Array of all child nodes of a node or an empty array if there are no child nodes.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>xml.Node[]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
<th>N/xml Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var childnodes = parentNode[0].childNodes;
...
//Add additional code
```
Node.firstChild

Property Description  
The first child node of a node, or null if there are no child nodes.

Type  
xml.Node

Supported Script Types  
All script types

Module  
N/xml Module

Since  
2015.2

Syntax

```javascript
//Add additional code
...
var nodeValue1 = bookNode[0].firstChild.nextSibling.textContent;
...
//Add additional code
```

Node.lastChild

Property Description  
The last child node of a node, or null if there are no child nodes.

Type  
xml.Node

Supported Script Types  
All script types

Module  
N/xml Module

Since  
2015.2

Syntax

```javascript
//Add additional code
...
var nodeValue = parentNode[0].lastChild.previousSibling.textContent;
...
//Add additional code
```

Node.localName

Property Description  
The local part of the qualified name of a node.

Type  
string (read-only)

Supported Script Types  
All script types
<table>
<thead>
<tr>
<th>Module</th>
<th>N/xml Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var localname = parentNode[0].localName;
...
//Add additional code
```

**Node.namespaceURI**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The namespace uniform resource identifier (URI) of a node or <strong>null</strong> if there is no namespace URI for the node.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <em>SuiteScript 2.0 Script Types.</em></td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var uri = parentNode[0].namespaceURI;
...
//Add additional code
```

**Node.nextSibling**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The next node in a node list or <strong>null</strong> if the current node is the last node.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>xml.Node (read-only)</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td></td>
<td>For more information, see the help topic <em>SuiteScript 2.0 Script Types.</em></td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
```
Node.nodeName

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Name of a node, depending on the type. For example, for a node of type <code>xml.Element</code>, the name is the name of the element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic [SuiteScript 2.0 Script Types](#). |
| Module               | N/xml Module                                                                                    |
| Since                | 2015.2                                                                                          |

**Syntax**

```javascript
//Add additional code
...
var nodeName = parentNode[0].firstChild.nextSibling.textContent;
...
//Add additional code
```

Node.nodeType

| Property Description | The type of node as an enum.  
For all possible values of this property, see `xml.NodeType`. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td><code>xml.NodeType</code></td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic [SuiteScript 2.0 Script Types](#). |
| Module               | N/xml Module                                                                                    |
| Since                | 2015.2                                                                                          |

**Syntax**

```javascript
//Add additional code
...
var nodeType = parentNode[0].firstChild.nodeType;
...
//Add additional code
```
Node.nodeValue

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The value of a node, depending on its type. If the value is null, setting this value has no effect.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Module               | N/xml Module                                                                                     |
| Since                | 2015.2                                                                                           |

Syntax

```javascript
//Add additional code
...
var nodeValue = parentNode[0].firstChild.nodeValue;
...
//Add additional code
```

Node.ownerDocument

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The root element for a node as a xml.Document object. Use this object to create new nodes with Document.createElement(options) or Document.createElementNS(options).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>xml.Document</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Module               | N/xml Module                                                                                                                          |
| Since                | 2015.2                                                                                                                                   |

Syntax

```javascript
//Add additional code
...
var doc = parentNode[0].ownerDocument;
...
//Add additional code
```

Node.parentNode

| Property Description | The parent node of a node. All node types, except xml.Attr, xml.Document, DocumentFragment, Entity, and Notation can have a parent node.  
See xml.NodeType for possible node types. |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>xml.Node</td>
</tr>
</tbody>
</table>
### Node.prefix

**Property Description**
The namespace prefix of the node, or `null` if the node does not have a namespace. If the value is `null`, setting it has no effect, including read-only node types.

**Type**
`string`

**Supported Script Types**
All script types

---

**Module**
N/xml Module

**Since**
2015.2

---

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>NAMESPACE_ERR: An attempt is made to create or change an object in a way which is incorrect with regard to namespaces.</td>
<td>Cannot edit the node prefix.</td>
</tr>
</tbody>
</table>

---

### Syntax

```javascript
//Add additional code ...
var nodeValue = parentNode[0].lastChild.parentNode.textContent;
...
//Add additional code
```

---

### Node.previousSibling

**Property Description**
The previous node in a node list or `null` if the current node is the first node.

**Type**
`xml.Node`

**Supported Script Types**
All script types

---

**Module**
N/xml Module

**Since**
2015.2
### N/xml Module

For more information, see the help topic SuiteScript 2.0 Script Types.

<table>
<thead>
<tr>
<th>Module</th>
<th>N/xml Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
var nodeValue = parentNode[0].lastChild.previousSibling.textContent;
...
//Add additional code
```

### Node.textContent

<table>
<thead>
<tr>
<th>Property Description</th>
<th>The textual content of a node and its descendants. If the value is <code>null</code>, then setting it has no effect. If you set this value, any child nodes are removed and replaced by a single text node with this string as a value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types  For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
var nodeValue = parentNode[0].firstChild.textContent;
...
//Add additional code
```

### xml.Document

<table>
<thead>
<tr>
<th>Object Description</th>
<th>Represents an entire XML document. The XML DOM presents a document as a hierarchy of node objects. Use the methods and properties available to the xml.Document object to manipulate the XML document and the nodes in the document tree. For a list of this object's methods and properties, see Document Object Members. An XML document object is also a node of type DOCUMENT_NODE. In addition to the Document object members, Document objects inherit the members of the Node object. For a complete list of these methods and properties, see Node Object Members.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
</tbody>
</table>
## Document.adoptNode(options)

### Method Description
Attempts to adopt a node from another document to this document. If successful, this method changes the Node.ownerDocument property of the source node, its children, and any attribute nodes to the current document. If the source node has a parent node, the parent node is first removed from the child list of its own parent node.

**Important:** This method is not supported on Internet Explorer.

### Returns
xml.Node

### Supported Script Types
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

### Governance
None

### Module
N/xml Module

### Since
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.source</td>
<td>xml.Node</td>
<td>Required</td>
<td>Source node to add as a child into the current node object.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>NOT_FOUND_ERR: An attempt is made to reference a node in a context where it does not exist.</td>
<td>Node cannot be adopted.</td>
</tr>
</tbody>
</table>
Document.createAttribute(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates an attribute node of type ATTRIBUTE_NODE with the optional specified value and returns the new xml.Attr object. The localName, prefix, and namespaceURI properties of the new node are set to null.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>xml.Attr</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Name of the new attribute node.</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>Optional</td>
<td>Value for the attribute node. If unspecified, the value is an empty string.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.</td>
<td>Attribute with the specified name or value cannot be created.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var attr = xmlDocument.createAttribute({
    name: 'lang',
});
... //Add additional code
```
Document.createAttributeNS(options)

**Method Description**

Creat**es an attribute node of type ATTRIBUTE_NODE, with the specified namespace value and optional specified value, and returns the new xml.Attr object.**

The Node.localName, Node.prefix, and Node.namespaceURI properties of the new node are set to null.

**Important:** This method is not supported on Internet Explorer.

<table>
<thead>
<tr>
<th>Returns</th>
<th>xml.Attr</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
<td></td>
</tr>
<tr>
<td><strong>Governance</strong></td>
<td>None</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/xml Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI of the attribute to create. Value can be null.</td>
</tr>
<tr>
<td>options.qualifiedName</td>
<td>string</td>
<td>Required</td>
<td>Qualified name of the new attribute node.</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>Optional</td>
<td>Value for the attribute node. If unspecified, the value is an empty string.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.</td>
<td>Attribute with the specified value cannot be created.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var attr = xmlDocument.createAttributeNS({
    namespaceURI : '*',
    ...});
```
Document.createCDATASection(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a CDATA section node of type DOCUMENT_FRAGMENT_NODE with the specified data and returns the new xml.Node object.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>xml.Node</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None                                                                                                           |
| Module             | N/xml Module                                                                                                   |
| Since              | 2015.2                                                                                                         |

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.data</td>
<td>string</td>
<td>Required</td>
<td>Data for the new CDATA section node.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: data</td>
<td>Cannot create CDATA section node with the specified data.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var newNode = xmlDocument.createCDATASection({
  data : 'Limited Edition.'
});
...
//Add additional code
```

Document.createComment(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Creates a Comment node of type COMMENT_NODE with the specified string.</th>
</tr>
</thead>
</table>

SuiteScript 2.0 API Reference
## xml Node

### Returns
xml.Node

### Supported Script Types
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

### Governance
None

### Module
N/xml Module

### Since
2015.2

### Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.data</td>
<td>string</td>
<td>Required</td>
<td>Data for the Comment node.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
var newNode = xmlDocument.createComment({
  data : 'This is a comment.'
});
...
//Add additional code
```

## Document.createDocumentFragment()

### Method Description
Creates a node of type DOCUMENT_FRAGMENT_NODE and returns the new xml.Node object.

### Returns
xml.Node

### Supported Script Types
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

### Governance
None

### Module
N/xml Module

### Since
2015.2

### Syntax

```javascript
//Add additional code
...
var newNode = xmlDocument.createDocumentFragment();
...
//Add additional code
```
Document.createElement(options)

Method Description

Creates a new node of type ELEMENT_NODE with the specified name and returns the new xml.Element node.

The Node.localName, Node.prefix, and Node.namespaceURI properties of the new node are set to null.

Returns

xml.Element

Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/xml Module

Since

2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.tagName</td>
<td>string</td>
<td>Required</td>
<td>Name of the element to create.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.</td>
<td>Element cannot be created with the specified tagName value.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var elem = xmlDocument.createElement({
  tagName : 'book'
});
...
//Add additional code
```

Document.createElementNS(options)

Method Description

Creates a new node of type ELEMENT_NODE with the specified namespace URI and name and returns the new xml.Element object.

The Node.localName, Node.prefix, and Node.namespaceURI properties of the new node are set to null.
XML Element Creation

**Module**
N/xml Module

**Supported Script Types**
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Since**
2015.2

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI of the element to create. Can be null.</td>
</tr>
<tr>
<td>options.qualifiedName</td>
<td>string</td>
<td>Required</td>
<td>Qualified name of the element to create.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.</td>
<td>Element with the specified namespace cannot be created.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var elem = xmlDocument.createElementNS(
   namespaceURI : '*',
   qualifiedName : 'book'
);
...  
//Add additional code
```

**Method Description**

`Document.createProcessingInstruction(options)`

Creates a new node of type PROCESSING_INSTRUCTION_NODE with the specified target and data and returns the new `xml.Node` object.

The following example shows a sample processing instruction:

```xml
<?xml version="1.0"?>
```

Use a processing instruction node to keep processor-specific information in the text of the XML document.

**Returns**

xml.Node
**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.target</td>
<td>string</td>
<td>Required</td>
<td>Target part of the processing instruction.</td>
</tr>
<tr>
<td>options.data</td>
<td>string</td>
<td>Required</td>
<td>Data for the processing instruction.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.</td>
<td>Processing instruction node cannot be created with the specified target or data.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var newNode = xmlDocument.createProcessingInstruction(
    target : 'xml'
    data : 'version="1.0"'
);  
... //Add additional code
```

**Document.createTextNode(options)**

**Method Description**

Creates a new text node and returns the new `xml.Node` object.

**Returns**

`xml.Node`

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/xml Module

**Since**

2015.2
Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.data</td>
<td>string</td>
<td>Required</td>
<td>Data for the text node.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var newNode = xmlDocument.createTextNode({
    data : 'Sample Title'
});
...
//Add additional code
```

Document.getElementById(options)

Method Description

Returns the element that has an ID attribute with the specified value as an xml.Element object. Returns null if no such element exists.

Returns

xml.Element

Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/xml Module

Since

2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.elementId</td>
<td>string</td>
<td>Required</td>
<td>Unique ID value for an element.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var elem = xmlDocument.getElementById({
    elementId : 'id12345'
});
...
//Add additional code
```
**Document.getElementsByTagName(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns an array of xml.Element objects with a specific tag name, in the order in which they appear in the XML document.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>xml.Element[]</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None                                                                                           |
| Module             | N/xml Module                                                                                 |
| Since              | 2015.2                                                                                         |

**Parameters**

ℹ️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.tagName</td>
<td>string</td>
<td>Required</td>
<td>Case-sensitive tag name of the element to match on. Use the * wildcard to match all elements.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var elem = xmlDoc.getElementsByTagName({
  tagName : 'book'
});
...
//Add additional code
```

**Document.getElementsByTagNameNS(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns an array of xml.Element objects with a specific tag name and namespace, in the order in which they appear in the XML document.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>xml.Element[]</td>
</tr>
</tbody>
</table>
| Supported Script Types | All script types  
For more information, see the help topic SuiteScript 2.0 Script Types. |
| Governance         | None                                                                                           |
| Module             | N/xml Module                                                                                 |
| Since              | 2015.2                                                                                         |
Parameters

### Note:
The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI to match on. Use the * wildcard to match all namespaces.</td>
</tr>
<tr>
<td>options.localName</td>
<td>string</td>
<td>Required</td>
<td>Localname property to match on. Use the * wildcard to match all local names.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var elem = xmlDocument.getElementsByTagNameNS({
  namespaceURI : '*',
  localName : 'book'
});
...
//Add additional code
```

**Document.importNode(options)**

**Method Description**

Imports a node from another document to this document. This method creates a new copy of the source node.

- If the `deep` parameter is set to `true`, it imports all children of the specified node.
- If set to `false`, it imports only the node itself.

Method returns the imported `xml.Node` object.

**Returns**

`xml.Node`

**Supported Script Types**

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](https://docs.oracle.com/ netlilux/19.2/whats-new/topic/suiteapp-developer-guide/script-types.html).

**Governance**

None

**Module**

N/xml Module

**Since**

2015.2

Parameters

### Note:
The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.importedNode</td>
<td><code>xml.Node</code></td>
<td>Required</td>
<td>Node from another XML document to import.</td>
</tr>
<tr>
<td>options.deep</td>
<td><code>boolean</code></td>
<td><code>true</code></td>
<td>Required</td>
</tr>
</tbody>
</table>
### Document.doctype

**Property Description**

The doctype of the XML document.

**Type**

`xml.Element (read-only)`

**Supported Script Types**

All script types

For more information, see the help topic **SuiteScript 2.0 Script Types.**

**Module**

N/xml Module

**Since**

2015.2

**Syntax**

```javascript
//Add additional code
...
var doctype = xmlDocument.doctype;
...  
//Add additional code
```

### Document.documentElement

**Property Description**

Root node of the XML document.
Use this property to directly access the `xml.Element` object that represents the root node of an XML document.

**Type**  
`xml.Element` (read-only)

**Supported Script Types**  
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  
N/xml Module

**Since**  
2015.2

**Syntax**

```javascript
//Add additional code
...
var root = xmlDocument.documentElement;
...
//Add additional code
```

---

**Document.documentURI**

**Property Description**  
Location of the document or `null` if undefined.

**Type**  
string

**Supported Script Types**  
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  
N/xml Module

**Since**  
2015.2

**Syntax**

```javascript
//Add additional code
...
var documentURI = xmlDocument.documentURI;
...
//Add additional code
```

---

**Document.inputEncoding**

**Property Description**  
Encoding used for an XML document at the time the document was parsed.

When parsing an XML document with the following declaration, the `inputEncoding` property is UTF-8:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

⚠️ **Important:** The value of this property is browser-specific.

**Type**  
string (read-only)
**N/xml Module**

**Supported Script Types**  
All script types  
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  
N/xml Module

**Since**  
2015.2

**Syntax**

```javascript
//Add additional code
...
var encoding = xmlDocument.inputEncoding;
...
//Add additional code
```

### Document.xmlEncoding

**Property Description**  
Part of the XML declaration, the XML encoding of the XML document.  
In the following declaration, the xmlEncoding property is UTF-8:  
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

**Important:** This property is not supported on Internet Explorer or Firefox.

**Type**  
string (read-only)

**Supported Script Types**  
All script types  
For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**  
N/xml Module

**Since**  
2015.2

**Syntax**

```javascript
//Add additional code
...
var encoding = xmlDocument.xmlEncoding;
...
//Add additional code
```

### Document.xmlStandalone

**Property Description**  
Part of the XML declaration, returns `true` if the current XML document is standalone or returns `false` if it is not.  
In the following declaration, the xmlStandalone property is `true`:  
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```
**Important:** This property is not supported on Internet Explorer or Firefox.

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>boolean true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
<td>For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/xml Module</td>
<td></td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
<td></td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var isStandalone = xmlDocument.xmlStandalone;
...
//Add additional code
```

### Document.xmlVersion

**Property Description**

Part of the XML declaration, the version number of the XML document.

In the following declaration, the xmlVersion property is 1.0:

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
```

**Important:** This property is not supported on Internet Explorer or Firefox.

<table>
<thead>
<tr>
<th><strong>Type</strong></th>
<th>string (read-only)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Supported Script Types</strong></td>
<td>All script types</td>
</tr>
<tr>
<td><strong>Module</strong></td>
<td>N/xml Module</td>
</tr>
<tr>
<td><strong>Since</strong></td>
<td>2015.2</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th><strong>Error Code</strong></th>
<th><strong>Message</strong></th>
<th><strong>Thrown If</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>The implementation does not support the requested type of object or operation.</td>
<td>Cannot edit the XML version for the document.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
var version = xmlDocument.xmlVersion;
...
```
xml.Element

**Object Description**

Represents an element in an XML document. Elements may contain attributes, other elements, or text. If an element contains text, the text is represented in a text node of type TEXT_NODE.

For example, the following element year contains a text node with the value of 2015:

```xml
<year>2015</year>
```

For a list of this object's methods and properties, see [Element Object Members](#).

An XML element object is also a node of type ELEMENT_NODE. In addition to the Element object members, Element objects inherit the members of the Node object. For a complete list of these methods and properties, see [Node Object Members](#).

### Supported Script Types

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

### Module

N/xml Module

### Since

2015.2

### Syntax

```javascript
//Add additional code
...
var elem = parentNode[0].getElementsByTagName('tagName');
...
//Add additional code
```

### Element.getAttribute(options)

**Method Description**

Returns the value of the specified attribute.

**Returns**

xml.Attr

**Supported Script Types**

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

None

**Module**

N/xml Module

**Since**

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Name of the attribute for which to return the value.</td>
</tr>
</tbody>
</table>
Element.getAttributeNode(options)

Method Description
Retrieves an attribute node by name.

Important: This method is not supported on Internet Explorer.

Returns
xmlAttr

Supported Script Types
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/xml Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>The name of the attribute to return.</td>
</tr>
</tbody>
</table>

Syntax

//Add additional code
...
var attr = elem[0].getAttributeNode({
   name : 'lang'
});
...
//Add additional code

Element.getAttributeNodeNS(options)

Method Description
Returns an attribute node with the specified namespace URI and local name.
## Element.getAttributeNS(options)

### Returns

Returns a string.

### Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

### Governance

None

### Module

N/xml Module

### Since

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI of the attribute to return. Value can be <code>null</code>.</td>
</tr>
<tr>
<td>options.localName</td>
<td>string</td>
<td>Required</td>
<td>Local name of the attribute to return.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: namespaceURI</td>
<td>Attribute node with the specified namespace cannot be retrieved.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
var attr = elem[0].getAttributeNS({
    namespaceURI : '',
    localName : 'lang'
});
...
//Add additional code
```

### Element.getAttributeNS(options)

- **Method Description**
  
  Returns an attribute value with the specified namespace URI and local name.

  **Important:** This method is not supported on Internet Explorer.

- **Returns**
  
  String
### Supported Script Types
All script types
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

### Governance
None

### Module
N/xml Module

### Since
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI of the attribute to return. Value can be null.</td>
</tr>
<tr>
<td>options.localName</td>
<td>string</td>
<td>Required</td>
<td>Local name of the attribute to return.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: namespaceURI</td>
<td>Attribute with the specified namespace cannot be retrieved.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
var attr = elem[0].getAttributeNS({
    namespaceURI : '*',
    localName : 'lang'
});
...
//Add additional code
```

### `Element.getElementsByTagName(options)`

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns</th>
<th>Supported Script Types</th>
<th>Governance</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns an array of descendant <code>xmlElement</code> objects with a specific tag name, in the order in which they appear in the XML document.</td>
<td><code>xmlElement[]</code></td>
<td>All script types</td>
<td>None</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.tagName</td>
<td>string</td>
<td>Required</td>
<td>Case-sensitive tag name of the element to match on. Use the * wildcard to match all elements.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var elem = parentNode[0].getElementsByTagName({tagName : 'title'});
...
//Add additional code
```

**Element.getElementsByTagNameNS(options)**

**Method Description**

Returns an array of descendant `xml.Element` objects with a specific tag name and namespace, in the order in which they appear in the XML document.

**Important:** This method is not supported on Internet Explorer.

**Returns**

`xml.Element[]`

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

None

**Module**

N/xml Module

**Since**

2015.2

Parameters

*Note:* The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI to match on. Use the * wildcard to match all namespaces.</td>
</tr>
<tr>
<td>options.localName</td>
<td>string</td>
<td>Required</td>
<td>Localname property to match on. Use the * wildcard to match all local names.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: namespaceURI</td>
<td>Elements with the specified namespace cannot be retrieved.</td>
</tr>
</tbody>
</table>
Syntax

```javascript
// Add additional code
...
var elem = parentNode[0].getElementsByTagNameNS({
    namespaceURI : '*',
    localName : 'lang'
});
...
// Add additional code
```

**Element.hasAttribute**(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Returns true if the current element has an attribute with the specified name or if that attribute has a default value. Otherwise, returns false.</th>
</tr>
</thead>
</table>

**Important:** This method is not supported on Internet Explorer.

**Returns**

| boolean true | false |

**Supported Script Types**

| All script types |

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**

| None |

**Module**

| N/xml Module |

**Since**

| 2015.2 |

**Parameters**

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Name of the attribute to match on.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
// Add additional code
...
var attrExists = elem[0].hasAttribute({
    name : 'lang'
});
...
// Add additional code
```

**Element.hasAttributeNS**(options)

| Method Description | Returns true if the current element has an attribute with the specified local name and namespace or if that attribute has a default value. Otherwise, returns false. |

**Returns**

| boolean true | false |

**Supported Script Types**

| All script types |

For more information, see the help topic SuiteScript 2.0 Script Types.
Important: This method is not supported on Internet Explorer.

Returns

| boolean | true | false |

Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None

Module

N/xml Module

Since

2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI of the attribute to match on.</td>
</tr>
<tr>
<td>options.localName</td>
<td>string</td>
<td>Required</td>
<td>Local name of the attribute to match on.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: namespaceURI</td>
<td>The method is called with an illegal namespace value.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var attrExists = elem[0].getAttributeNS({
  namespaceURI : '*',
  localName : 'lang'
});
...
//Add additional code
```

Element.removeAttribute(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Removes the attribute with the specified name.</th>
</tr>
</thead>
</table>

Returns

| void |

Supported Script Types

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Governance

None
Module | N/xml Module
---|---
Since | 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.name</td>
<td>string</td>
<td>Required</td>
<td>Name of the attribute to remove.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: name</td>
<td>Attribute with the specified name cannot be removed.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
elem[0].removeAttribute({
    name : 'lang'
});
...
//Add additional code
```

**Element.removeAttributeNode(options)**

Method Description

Removes the attribute specified as a `xml.Attr` object.

Returns

`xml.Attr`

Supported Script Types

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

Governance

None

Module | N/xml Module
---|---
Since | 2015.2

Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.oldAttr</td>
<td><code>xml.Attr</code></td>
<td>Required</td>
<td><code>xml.Attr</code> object to remove.</td>
</tr>
</tbody>
</table>
Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>NOT_FOUND_ERR: An attempt is made to reference a node in a context where it does not exist.</td>
<td>Attribute node cannot be removed.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
var removedAttr = elem[0].removeAttributeNode({
  oldAttr : attr
});
...
//Add additional code
```

**Element.removeAttributeNS(options)**

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Removes the attribute with the specified namespace URI and local name.</th>
</tr>
</thead>
</table>

⚠️ **Important:** This method is not supported on Internet Explorer.

**Returns**

void

**Supported Script Types**

All script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

None

**Module**

N/xml Module

**Since**

2015.2

**Parameters**

ℹ️ **Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI of the attribute node to remove.</td>
</tr>
<tr>
<td>options.localName</td>
<td>string</td>
<td>Required</td>
<td>Local name of the attribute node to remove.</td>
</tr>
</tbody>
</table>

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: namespaceURI</td>
<td>Attribute with the specified namespace cannot be removed.</td>
</tr>
</tbody>
</table>
Element.setAttribute(options)

**Method Description**
Adds a new attribute with the specified name. If an attribute with that name is already present in the element, its value is changed to the value specified in method argument.
If an attribute with the specified name already exists, the value of the attribute is changed to the value of the value parameter.

**Returns**
void

**Supported Script Types**
All script types
For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**
None

**Module**
N/xml Module

**Since**
2015.2

**Parameters**

- **options.name** (string) **Required**
  Name of the attribute to add.

- **options.value** (string) **Required**
  Value of the attribute to add.

**Errors**

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.</td>
<td>Value for the attribute cannot be set.</td>
</tr>
</tbody>
</table>

**Syntax**

```javascript
//Add additional code
...
```

```javascript
elem[0].removeAttributeNS({
    namespaceURI : '',
    localName : 'lang'
});
...
//Add additional code
```
### Element.setAttributeNode(options)

<table>
<thead>
<tr>
<th>Method Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adds the specified attribute node. If an attribute with the same name is already present in the element, it is replaced by the new one. If an attribute with the same nodeName property already exists, it is replaced with the object in the newAttr parameter. If the attribute node replaces an existing attribute node, the method returns the new xml.Attr object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>xml.Attr</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Supported Script Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>All script types For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Governance</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/xml Module</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015.2</td>
</tr>
</tbody>
</table>

#### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.newAttr</td>
<td>xml.Attr</td>
<td>Required</td>
<td>New xml.Attr object to add to the xml.Element object.</td>
</tr>
</tbody>
</table>

#### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INUSE_ATTRIBUTE_ERR: An attempt is made to add an attribute that is already in use elsewhere.</td>
<td>Attribute node cannot be added.</td>
</tr>
</tbody>
</table>

#### Syntax

```javascript
//Add additional code
...
elem[0].setAttributeNode({
    newAttr : attr
});
...
//Add additional code
```
Element.setAttributeNodeNS(options)

Method Description
Adds the specified attribute node. If an attribute with the same local name and namespace URI is already present in the element, it is replaced by the new one. If an attribute with the same namespaceURI and localName property already exist, it is replaced with the object in the newAttr parameter. If the attribute node replaces an existing attribute node, the method returns the new xml.Attr object.

Important: This method is not supported on Internet Explorer.

Returns
xml.Attr

Supported Script Types
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Governance
None

Module
N/xml Module

Since
2015.2

Parameters

Note: The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.newAttr</td>
<td>xml.Attr</td>
<td>Required</td>
<td>New xml.Attr object to add to the xml.Element object.</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>INUSE_ATTRIBUTE_ERR: An attempt is made to add an attribute that is already in use elsewhere.</td>
<td>Attribute node cannot be added.</td>
</tr>
</tbody>
</table>

Syntax

```javascript
//Add additional code
...
elem[0].setAttributeNodeNS({
    newAttr : attr
});
...
//Add additional code
```

Element.setAttributeNS(options)

Method Description
Adds a new attribute with the specified name and namespace URI. If an attribute with the same name and namespace URI is already present in the element, its value is changed to the value specified in method argument.
If an attribute with the specified name already exists, the value of the attribute is changed to the value of the value parameter. If the attribute node replaces an existing attribute node, the method returns the new xmlAttr object.

**Important:** This method is not supported on Internet Explorer.

**Returns**
void

**Supported Script Types**
All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Governance**
None

**Module**
N/xml Module

**Since**
2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.namespaceURI</td>
<td>string</td>
<td>Required</td>
<td>Namespace URI of the attribute node to add.</td>
</tr>
<tr>
<td>options.qualifiedName</td>
<td>string</td>
<td>Required</td>
<td>Fully qualified attribute name to add.</td>
</tr>
<tr>
<td>options.value</td>
<td>string</td>
<td>Required</td>
<td>String value of the attribute to add.</td>
</tr>
</tbody>
</table>

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS(XML_DOM_EXCEPTION)</td>
<td>INVALID_CHARACTER_ERR: An invalid or illegal XML character is specified.</td>
<td>Attribute node with the specified value cannot be added.</td>
</tr>
</tbody>
</table>

### Syntax

```javascript
//Add additional code
...
elem[0].setAttributeNS(
    namespaceURI : '*',
    qualifiedName : 'lang',
    value : 'fr'
);    
...  
//Add additional code
```

**Element.tagName**

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>The tag name of this xml.Element object.</td>
<td></td>
</tr>
<tr>
<td>Type</td>
<td>string (read-only)</td>
</tr>
</tbody>
</table>
xml Attr

Object Description
Represents an attribute node of an xml.Element object.
For a complete list of this object’s properties, see Attr Object Members.

Supported Script Types
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/xml Module

Since
2015.2

Syntax

```javascript
//Add additional code
...
var attr = elem[0].getAttributeNode({
    name : 'lang'
});
...
//Add additional code
```

Attr.name

Property Description
The name of an attribute.
This property is a qualified name if the Node.localName property for the parent xml.Element object is null.

Type
string (read-only)

Supported Script Types
All script types
For more information, see the help topic SuiteScript 2.0 Script Types.

Module
N/xml Module
Since 2015.2

Syntax

```javascript
//Add additional code
...
var attrName = attr.name; \ returns 'lang'.
...
//Add additional code
```

### Attr.ownerElement

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
<th>Type</th>
<th>Supported Script Types</th>
</tr>
</thead>
</table>
| xml.Element object that is the parent of the xml.Attr object. Value is null if the attribute is not used by an element. | | xml.Element (read-only) | All script types

**Important:** This property is not supported on Internet Explorer.

Module N/xml Module

Since 2015.2

Syntax

```javascript
//Add additional code
...
var attrElement = attr.ownerElement; \ returns the title element.
...
//Add additional code
```

### Attr.specified

<table>
<thead>
<tr>
<th>Property Description</th>
<th>Description</th>
<th>Type</th>
<th>Supported Script Types</th>
</tr>
</thead>
</table>
| Returns true if the attribute value is set in the parsed XML document, and false if it is a default value in a DTD or Schema. | boolean | true | All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

Module N/xml Module

Since 2015.2

Syntax

```javascript
//Add additional code
```
Attr.value

| Property Description | Value of an attribute. The value of the attribute is returned as a string. Character and general entity references are replaced with their values. For example, a character reference such as `&#160;` or an entity reference such as `&nbsp;` is replaced with a non-breaking space. |

**Note:** If you set this value, it creates a text node with the unparsed contents of the string, for example, any characters that an XML processor would recognize as markup are instead treated as literal text.

<table>
<thead>
<tr>
<th>Type</th>
<th>string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported Script Types</td>
<td>All script types. For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Module</td>
<td>N/xml Module</td>
</tr>
<tr>
<td>Since</td>
<td>2015.2</td>
</tr>
</tbody>
</table>

Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Message</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOM_EXCEPTION</td>
<td>Invalid argument type, expected string: value</td>
<td>Cannot set the attribute value with the specified value.</td>
</tr>
</tbody>
</table>

Syntax

```plaintext
//Add additional code
...
var attrValue = attr.value;
...
//Add additional code
```

xml.escape(options)

<table>
<thead>
<tr>
<th>Method Description</th>
<th>Prepares a string for use in XML by escaping XML markup, such as angle brackets, quotation marks, and ampersands.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Returns</td>
<td>string</td>
</tr>
<tr>
<td>Supported Script Types</td>
<td>All script types. For more information, see the help topic SuiteScript 2.0 Script Types.</td>
</tr>
<tr>
<td>Governance</td>
<td>None</td>
</tr>
</tbody>
</table>
## xml.escape

**Method Description**

Validates an XML document against an XML Schema (XSD).

**Important:** This method only validates XML Schema (XSD); validation of other XML schema languages is not supported.

The XML document must be passed as an `xml.Document` object. The location of the source XML Document does not matter; the validation is performed with the Document object stored in memory. The XSD must be stored in the File Cabinet.

**Returns**

`void`

**Supported Script Types**

All server-side script types

For more information, see the help topic [SuiteScript 2.0 Script Types](#).

**Governance**

None

**Module**

`N/xml Module`

**Since**

2015.2

### Parameters

**Note:** The options parameter is a JavaScript object.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Type</th>
<th>Required / Optional</th>
<th>Description</th>
<th>Since</th>
</tr>
</thead>
<tbody>
<tr>
<td>options.xml</td>
<td><code>xml.Document</code></td>
<td>Required</td>
<td>The <code>xml.Document</code> object to validate.</td>
<td>2015.2</td>
</tr>
</tbody>
</table>
### xml.validate

```javascript
// Add additional code
...
xml.validate({
    xml: xmlDocument,
    xsdFilePathOrId: 'SuiteScripts/schema_parent.xsd',
    importFolderPathOrId: 'SuiteScripts/'
});
...
// Add additional code
```

### xml.nodeType

**Enum Description**

Enumeration that holds the string values for the supported node types. The `Node.nodeType` property is defined by one of the values in this enum. Use this enum to determine the type of a node in an XML document.

**Note:** Enum values are constants and therefore read-only.

**Supported Script Types**

All script types

For more information, see the help topic SuiteScript 2.0 Script Types.

**Module**

N/xml Module

**Since**

2015.2

**Values**

- ATTRIBUTE_NODE
- CDATA_SECTION_NODE
- DOCUMENT_NODE
- DOCUMENT_TYPE_NODE
- ENTITY_REFERENCE_NODE
- NOTATION_NODE

### Errors

<table>
<thead>
<tr>
<th>Error Code</th>
<th>Thrown If</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSS_XML_DOES_NOT_CONFORM_TO_SCHEMA</td>
<td>The provided XML is invalid for the provided schema.</td>
</tr>
<tr>
<td>SSS_INVALID_XML_SCHEMA_OR_DEPENDENCY</td>
<td>Schema is an incorrectly structured XSD or the dependent schema cannot be found.</td>
</tr>
</tbody>
</table>

### 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>options.xsdFilePathOrId</td>
<td>number</td>
<td>string</td>
<td>Required</td>
<td>The file ID or path to the XSD in the File Cabinet to validate the XML document against.</td>
</tr>
<tr>
<td>options.importFolderPathOrId</td>
<td>number</td>
<td>string</td>
<td>Optional</td>
<td>The folder ID or path to a folder in the File Cabinet containing additional XSD schemas which are imported by the parent XSD.</td>
</tr>
</tbody>
</table>
### Syntax

```javascript
//Add additional code
...
var DocType = xmlDocument.nodeType; // returns DOCUMENT_NODE
...
//Add additional code
```